IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s): L.D. Bergman et al.
Case: YOR920000742US1
Serial No.: 09/923,530
Filing Date: August 7, 2001
Group: 2162
Examiner: Baoquoc N. To

Title: Methods and Apparatus for Indexing Data in a Database and for Retrieving Data From a Database in Accordance With Queries Using Example Sets

---

DECLARATION OF PRIOR INVENTION UNDER 37 C.F.R. §1.131

We, the undersigned, hereby declare and state as follows:

1. We are the named inventors on the above-referenced U.S. patent application.

2. We conceived the invention that is the subject of the present application at least as early as October 30, 2000. On or about this date, an IBM disclosure document for an invention entitled "Indexing Method For Queries Using Multiple Positive and Negative Examples" was sent to the inventors' attorneys at the law firm of Ryan, Mason & Lewis, LLP, for preparation of a related patent application. The accompanying letter, dated October 30, 2000, from IBM in-house counsel David M. Shofi and the IBM disclosure document are attached hereto as Exhibit 1.

3. The IBM disclosure document was written by inventor Vittorio Castelli.

4. Subsequent to conception and on or about January 2, 2001, inventor Vittorio Castelli sent a draft patent application and relative figures for the invention that is the subject of the present

application to the inventors' attorneys at the law firm of Ryan, Mason & Lewis, LLP, to assist in the preparation of the related patent application. The accompanying letter, dated January 2, 2001, from inventor Vittorio Castelli, and the draft patent application are attached hereto as Exhibit 2.

5. On or about July 23, 2001, inventor Vittorio Castelli received a draft patent application from inventors' attorneys at Ryan, Mason & Lewis, LLP. The fax transmission cover sheet is attached hereto as Exhibit 3.

4. Due diligence was performed in the preparation and filing of a patent application from the date the letter and IBM disclosure document were received until the application was filed on August 7, 2001.

5. All statements made herein of our own knowledge are true, and all statements made on information and belief are believed to be true.

6. We understand that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001, and may jeopardize the validity of the application or any patent issuing thereon.

Date: _____ _____

Lawrence D. Bergman

Date: _____ _____

Vittorio Castelli

Date: 02/15/2005 _____

Chung-Sheng Li

IBM

Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

October 30, 2000

VIA OVERNIGHT MAIL

RECEIVED
WITH THANKS
RYAN, MASON & LEWIS, LLP
11-1-00

William E. Lewis, Esq.
Ryan, Mason & Lewis, LLP
90 Forrest Avenue
Locust Valley, New York 11560

-177 Subject: Preparation of Patent Application: YOR920000742US1
Yorktown Disclosure Number: YOR8-2000-0440

Title: "INDEXING METHOD FOR QUERIES USING MULTIPLE POSITIVE AND NEGATIVE EXAMPLES "

Inventor:				
	Lawrence Bergman	1-914-784-7946	1N-D20	Hawthorne
	Vittorio Castelli	1-914-945-2396	36-106D	Yorktown
	Chung-Sheng Li	1-914-784-6661	2S-D62	Hawthorne

Dear Bill,

Further to our telephone conversation of last week, enclosed are materials relative to the preparation and prosecution of the subject patent application including an original disclosure, an embodiment, 13 prior art publications and a diskette with a soft copy of the embodiment text in Lotus Word Pro format and figures in Lotus Freelance format.

The formal papers are to be prepared and filed by your office, listing the names of the Yorktown attorneys on the Declaration and Power of Attorney as follows:

POWER OF ATTORNEY: As a named inventor I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith as follows:

Manny W. Schecter (Reg. 31,722), Lauren C. Bruzzone (Reg. 35,082), Christopher A. Hughes (Reg. 26,914), Edward A. Pennington (Reg. 32,588), John E. Hoel (Reg. 26,279), Joseph C. Redmond, Jr. (Reg. 18,753), Richard M. Ludwin (Reg. 33,010), Marc A. Ehrlich (Reg. 39,966), Wayne L. Ellenbogen(Reg. 43,602), Stephen C. Kaufman (Reg 29,551), Marian Underweiser (Reg. 46,134), David M. Shofi (Reg. 39, 835), Robert M. Trepp (Reg 25,933), Louis P. Herzberg (Reg. 41,500), Louis J. Percello (Reg. 33,206), Paul J. Otterstedt (Reg. 37,411), Daniel P. Morris (Reg. 32,053), and Douglas W. Cameron (Reg 31,596).

Send correspondence to: outside counsel attorney
Direct Telephone Calls to: outside counsel attorney

After the formal papers have been prepared by your office, please send them directly to the inventors for signature with a request that they sign all the forms. As you know, notarization is no longer required by Yorktown. Please instruct the inventors to return the executed formal papers directly to your office.

Please be advised that an additional step in our procedure is required when filing all original IBM Yorktown Patent Applications in the USPTO. The additional step is that a "Taiwan Oath & Assignment" form must be completed. The form must have all the required information completely filled in and must be signed and dated by all named inventors in the subject patent application. The second page of the two page form has a section entitled, "Note 1", which contains an alphabetized code reference depicting what information must be entered into each corresponding letter field (example: (a), (b), etc.). For your convenience, enclosed on the diskette is a soft copy of the "Taiwan Oath & Assignment" form to retain for continued use when preparing all new patent applications. When the form is complete, and all inventor(s) signatures and dates have been obtained, please forward the original Taiwan Oath & Assignment back to my office.

An Associate Power of Attorney form should be prepared and forwarded to this office for signature.

Before filing, please send me, by facsimile or otherwise, a copy of the claims if not the application as a whole, for my review. Furthermore, please send me a copy of the application (hard-copy and soft-copy (WordPro97 preferable)) and the formal papers, as filed, by express mail concurrently with your filing of the application. IBM will handle any US Maintenance fee payments internally.

Please conduct the work directly with the inventors listed above, including visiting as the situation warrants. The contact inventor is Vittorio Castelli.

Please inform this office on all points involving scope of coverage and finances. Please have your illustrator prepare formal drawings for the application. In the event you file with informal drawings, please provide us with formal drawings within three months of the filing date. If you have any questions contact me at the telephone number listed below.

Best regards,

David M. Shofi
Phone: (914) 945-3247
Fax: (914) 945-3281

/jd
Enclosures

Cc:
Lauren C. Bruzzone
Barbara Rasa
Lawrence Bergman
Vittorio Castelli
Chung-Sheng Li

Note that this application involves an invention developed under govt contract.

Disclosure YOR8-2000-0440

Created By: Vittorio Castelli **Created On:** 06/02/2000 09:25:07 AM
Last Modified By: Wendy R Petrovich **Last Modified On:** 06/05/2000 08:40:55 AM

*** IBM Confidential ***

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

Summary

Status	Under Evaluation
Processing Location	YOR
Functional Area	800 Dean - Systems
Attorney/Patent Professional	David Shofi/Watson/IBM
IDT Team	David Shofi/Watson/IBM
Submitted Date	06/02/2000 10:12:14 AM
Owning Division [Select]	RES
	50
Incentive Program	
Lab	
Technology Code	

Inventors with Lotus Notes IDs

Inventors: Vittorio Castelli/Watson/IBM, lawrence bergman, Chung-Sheng Li/Watson/IBM@IBMUS

Inventor Name > denotes primary contact	Inventor Serial	Div/Dept	Manager Serial	Manager Name
> Castelli, Vittorio	707255	22/935E	632465	Franaszek, Peter A.
Bergman, Lawrence	660474	22/1GSA	994145	Smith, John R.
Li, Chung-Sheng	397997	22/9AUC	707138	Feldman, Stuart I.

Inventors without Lotus Notes IDs

IDT Selection

IDT Team:	Attorney/Patent Professional:
David Shofi/Watson/IBM	David Shofi/Watson/IBM

Response Due to IP&L : 07/05/2000

Main Idea

*Title of disclosure (in English)

Indexing method for queries using multiple positive and negative examples

*Idea of disclosure

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

In a computer system containing a database supporting similarity searches, where the user specifies similarity by providing positive

and negative examples, the search engine must translate the examples into a definition of similarity and search the repository for matches. There are at least two types of queries.

i. ``best-k-matches", where the search engine must return the k database items that more closely match the concept specified by the user.
ii. ``threshold" search, were the search engine returns all the items in the database that are more similar than a specified similarity level to the concept described by the user.

Often, the user can refine the definition of the concept by selecting good and bad examples among the returned results; this is called ``relevance feedback".
The system executes the query by creating a scoring function based on the data, and scoring the database elements.

No indexing schemes exist that support efficiently queries of type i. and ii. There are two solutions to this problem in the current art.

1. Scoring the entire database at query execution time. This solution is computationally expensive, and practically unacceptable for large databases.
2. Defining a set of ``representative" queries, and using them to modify the database in such a way that the user queries can be processed using known indexing structures. This approach is also in general not satisfactory because
   - one would have to create a new modification for each application scenario, and potentially for each user.
   - the number of "representative queries" required to modify the database appropriately could be very large.
   - the process of modifying the database using the queries is in general iterative, and hence computationally very expensive.

We propose a method for searching a static index using a branch-and-bound algorithm designed to support queries of type i. and ii specified by the user by means of positive and negative examples.

The index will be universal (i.e., it would be constructed once for a given database, and used for all used queries) and will support query-dependent scoring functions (generated from the positive and negative examples).

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?
The invention is a combination of
1. a scoring function
2. an indexing method that supports searches based on this scoring function.

Scoring Function
Our scoring function will be constructed as follow:

Let the example be a collection of M rows of the database, each having N columns. We define the column means of the example as the averages of the column values over all the samples in the example, and the column standard deviation as the standard deviation of the column values computed over all the samples in the example.

Distances are converted to scores by means of monotonically decreasing mapping functions, which have values between 0 and 1; for instance, a trapezoidal function.

When scoring an item in the database, the scores with respect to the positive and negative examples are computed, and aggregated: the maximum of the positive scores is computed, and the maximum of the negative scores is computed, and subtracted from 1. The minimum of the resulting values is then used as a score for the database object.

Indexing

Indexing algorithms supporting the above scoring function are divided into two parts: index construction and index usage. We are not concerned with index construction per se. Our invention relates to:

1. the modification of existing index methods (thus, it is very general)
2. the method for searching the modified index

The class of indexing structures to which the invention applies are based on recursive partitioning of the search space in such a way that each recursive step refines the current partition.
Such classes of methods include:

- the *R-Tree* and almost all the derived indexing structures;
- the *KD-Tree* and almost all the derived indexing structures;
- the *Quadtree* and almost all the derived indexing structures;

We describe the invention in the context of the *ordered partition*, where the database is partitioned along one dimension at a time, in a predefined number of equi-depth bins.

Modification of existing index methods

The modification to the indexing structure needed to support the method of this invention consists of dynamically attaching to each node of the tree information on the best possible score of any database item stored as a descendant of that node. For example:

1. the minimum possible distance from each positive example
2. the maximum possible distance form each negative example

Search Method

The search method, for the k-nearest-neighbor problem
consists of the following steps:

1. starting at the root, compute the score of each child of the root with respect to the whole set of examples
2. selecting the node with higher score, and breaking ties appropriately
3. recursively applying 1. and 2. until a terminal node is reached
4. searching exhaustively the terminal node, i.e., scoring all the items and inserting the k items with highest score in an ranked list
5. backtracking: visit the siblings of the current node whose score is higher than the lowest score in the ranked list, in descending order of score. Visiting means applying steps 1.,2. and 3.
6. when the last candidate sibling has been visited, repeat 5) starting from the parent of the current node, until the root is reached.

An alternative strategy consists of steps 2. to 6. with the following step substituted for step 1.

1. start at the root, for each positive example, compute the score of each child of the root with respect to that positive example and all negative examples, perform steps 2. to 6. before using the next positive example.

A further strategy consists of applying steps 1. to 3. to all the nodes of the tree except the leaves, thus computing a score for the leaves, and searching exhaustively the leaves in order of decreasing score using all positive and negative examples, until the score of next leaf is lower than the lowest score in the ordered list of results.

Some Details

We define the distance of a node with respect to an example as follows:

- The distance of a node from a positive example is the minimum possible distance of any item in the subtree rooted at the node.
- The distance of a node from a negative example is the maximum possible distance of any item in the subtree rooted at the node.

The score of a node is the score computed by applying the mapping function to the distances from all positive and negative examples.

Note that the distance of a node from a positive example is smaller than or equal to the smallest of the distances of the children of the node from the positive example. Similarly, the distance of a node from a negative example is larger than or equal to the largest of the distances of its children from the negative example. Hence, the score of a node is larger than or equal to the highestscore of its children.

Using the ordered partition, the distance between a node and an example can be quickly computed using the distance between the parent and the example, and the position of the separating point used to compute the recursive partition at the node.

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have those others solved it and does your solution differ and why is it better?
The problem of searching based on multiple examples is becoming the standard in multimedia databases. We do not know of any solution that uses indexes to solve the problem.

4. If the invention is implemented in a product or prototype, include technical details, purpose, disclosure details to others and the date of that implementation.
The invention has been implemented, however the code is just a research prototype: it has not been advertised internally, nor disclosed externally.

IBM
T. J. Watson Res. Ctr.
P.O. Box 218
Yorktown Heights, NY, 10598

Vittorio Castelli

Ph: (914) 945 2396
FAx: (914) 945 4426
email: vittorio@us.ibm.com

Tuesday, January 02, 2001

William E. Lewis

Ryan, Mason & Lewis, LLP
90 Forest Avenue
Locust Valley, NY 11560

Dear Bill,

Please find enclosed a disk with the draft of the patent application and the relative figures.

The files are in a password-protected .zip file.

Looking forward to hearing from you,

I remain,

Yours Truly

Vittorio Castelli

LATEST VERSION (JAN'01) FROM VITTORIO CASTELLI. (taken from diskette sent)

Cross References to Related Applications

Docket YOR919980220US1, Method and Apparatus for Similarity Retrieval from Iterative Refinement (filed, 01/26/1999)

Field of the Invention

The current invention is related to improved similarity retrieval from databases, where the query is specified by providing multiple example sets.

Background

In a computer system containing a database supporting similarity searches, the term *query-by-example* denotes a paradigm where the user specifies a query by providing examples [FlickOth95]. In a multimedia database, an example can be an existing image or video, or a sketch drawn by the user. In a traditional database, an example can be a particular record.

There are at least two types of queries that can be specified by means of an example:

- **best-k-matches**, where the search engine must return the k database items that more closely match the concept specified by the user. The search engine uses the example to compute quantities (features), that are compared to the features of the items stored in the database. In multimedia databases, typical features include color, texture and shape. Given two items in the database, the one with more similar features to the query is considered the better match.
- **threshold** search, were the search engine returns all the items in the database that are more similar than a specified similarity level to the concept described by the user. The difference from the previous type of query resides in the similarity function, which is known (to a certain extent) to the user.

Similarity search using a single example is only moderately effective. To improve retrieval performance, the user can provide multiple examples [WanOth98]. Multiple examples are used to estimate the relative importance of the different features used in the retrieval, which translates into giving different weights to different features while computing similarity. The distance between the query vector **x** and a target vector **y** is then computed as

$$d^2(\mathbf{x},\mathbf{y}) = \sum w_i \, (\mathbf{x}[i]-\mathbf{y}[i])^2 \,, \qquad (1)$$

where the sum is over the M features in the feature vectors, and the w_i are different weights (this equation is taught, for example, in [WanOth98], equation (6).

For instance, if there are N positive examples, having feature vectors $\mathbf{x}_1,\ldots, \mathbf{x}_N$, a possible choice of weight for the ith feature is $w[i] = [\, (\mathbf{x}_1[i]^2+\mathbf{x}_2[i]^2+\ldots+\mathbf{x}_N[i]^2) / N \,]^{-1/2}$, where $\mathbf{x}_j[i]$ is the value of the ith feature for the jth example.

These examples can be positive (examples of the desired content) or negative (examples of undesired content). [WanOth98] teaches how to use both positive and negative examples to compute the weights w_i. Figure 1 shows the contour lines of the resulting distance function in the 2-dimensional case, i.e., the lines having the same distance from the query. In the figure, positive examples are denoted by a 'x' and negative examples by a circle with a minus sign.

Other distance metrics that can be used instead of (1) are the weighted Lp distances, computed as

$$d^p(\mathbf{x},\mathbf{y}) = \left(\sum w_i \, (\mathbf{x}[i]-\mathbf{y}[i])^p\right)^{1/p}$$

and quadratic distances computed as

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T K^{-1} (\mathbf{x}-\mathbf{y})$$

where K is a non-singular, positive-definite matrix, the (-1) superscript denotes the matrix inverse operator, and the (T) superscript denotes transposition.

Most authors teach how to solve the similarity search problem in a classification-like setting: they divide the database into classes, by learning from the user's input, as in [BanhuOth98], [MaManj96], [CoxOth2000]. Static approaches can be used, where the user's feedback is not allowed. For example, Ma and Manjunath [MaManj96] teach how to partition the database into clusters using a neural network approach. The taught method requires to label each database entry with a class label, hence it is tantamount to providing all the entries in the database as examples. The method is static since it produces a fixed pseudo-distance on the search space. This pseudo-distance does not depend on the individual queries, and cannot be changed by a user. The computational cost of the approach is prohibitive for large databases. Dynamic approaches [BanhuOth98] learn on-line, i.e., at query time. Mixed approaches [WoodOth98] use both query-time and off-line learning.

In this patent, we are concerned with assigning scores to the database items in a way that depends on the user's query, and returning the k elements with higher score, where k is a parameter of the query. Scoring means assigning a numerical value, for instance between 0 and 1, to an element. The semantics of scoring values is the following: elements with higher score are better matches for the query. The construction of a scoring function is an estimation problem, similar to the estimation of a regression function. Hence, as discussed in [DevOth96] (for example, on page 12), deducing a scoring function is a more complex problem than solving a discrimination (i.e., classification) problem. The motivation for solving a scoring problem rather than a classification problem is the lack of evidence regarding the existence of semantic classes associated with similarity retrieval. In other words, there is no evidence that the user is interested in a particular semantic concept when retrieving data based on similarity. Inferring the existence of such classes from the user's example reduces the flexibility of the system. The approach is valuable only if the user ask over and over for the same content, which is a rare occurrence. The present patent therefore does not rely on the simplifying classification framework.

In the relevance-feedback literature, authors propose solutions on how to use sets of positive and negative examples that are iteratively provided as input to the system. They assume that the user provides a set of positive examples or a set of negative examples (or both) at each iteration. The area of relevance feedback has been studied for the past 30 years. There are two main categories of relevance-feedback techniques from the viewpoint of how the system deals with examples provided during different iterations: static query rewrite, where at each iteration the same weight are given to all the examples irrespective of when they were added to the query [YongOth98]; and time-weighted query rewrite, where the system gives more importance to the most recently provided examples. Relevance-feedback techniques can also be divided in two categories, depending on how the examples are used: techniques that use adaptive ranking functions use the positive and negative examples to modify the weights of distance functions, such as (1) or variations, and are suited for on-line searches [YongOth98]; techniques that perform feature-space warping actually change the structure of the search space in non-linear fashions, are computationally very expensive, and are best suited for off-line learning [CastOth98,LiOth98].

In general, there is therefore a need to allow the user to provide simultaneously multiple sets of positive and negative examples and to use them in an on-line setting..

The process of searching a database is complex and time consuming. Data structures, called indexes or indexing structures, are used to speed up the process. In particular, multidimensional access methods simultaneously index several variables, as described, for instance in [GaeGunt98]. Multidimensional indexing methods are used for point queries, range queries as taught in [GaeGunt98] and nearest-neighbor queries, as taught, for instance, in [KimPark86], but not similarity queries based on multiple example sets. There is therefore a need for indexing structures supporting similarity queries based on multiple positive and negative example sets.

Constructing and maintaining multidimensional indexing structures are computationally expensive tasks. Substantial advantages would therefore be gained if similarity search based on multiple example set and conventional queries (i.e., point queries, range queries or nearest-neighbor queries) could be supported by the same indexing structure. Further, since multidimensional indexing structures supporting point, range and nearest-neighbor queries are well understood and therefore optimized versions exist, it would be advantageous to have a method to perform similarity search based on multiple example sets using a available indexing structure that supports point, range or nearest-neighbor queries, and in particular using the most common such methods, for example, the R-Trees, the k-d Trees, the Quadtrees and derived structures, taught in Section 3 of [GaeGunt98].

The current invention contains a method for similarity searching a data base, where the user specifies the query by providing one or more positive example sets and zero or more negative example sets. The invention includes a method for constructing on-line a scoring function and a method for retrieving the desired similar records from a database using a multidimensional index.

Brief Description of the Drawings

Figure 1 depicts an example of a system having features of the present invention, where the user formulates queries in terms of multiple positive and negative example sets.

Figure 2 depicts graphically an example set by representing each example as a point in a 2-dimensional feature space, each coordinate of which corresponds to the value of a different attribute.

Figure 3 is an example of representative (characteristic) sample and dispersion characterization of the example set shown in Figure 2.

Figure 4 is an example of the construction of a scoring function comprising the steps of computing the characteristic sample, dispersion characterization, and using both to produce the scoring function.

Figure 5 illustrates desirable properties of the scoring function of an example set.

Figure 6 is an example of how to use the dispersion characterization and the characteristic sample of an example set to construct a scoring function by first constructing a distance and then converting the distance into a score.

Figure 7 contains two example of functions that convert distances into scores having features of the present invention.

Figure 8 illustrates several contour line of a scoring function constructed using the example set of Figure 2 and the procedures of Figure 4 and 6.

Figure 9 describes a procedure for constructing a scoring function by first constructing individual scoring functions for each of the positive and negative example sets and then combining them.

Figure 10 describes a procedure for constructing a scoring function by first constructing individual scoring functions for each of the positive and negative example sets, then jointly modifying them, and finally combining them.

Figure 11 is an example of contour plot of a scoring function obtained from three positive example sets and two negative example sets using the procedure of Figure 10.

Figure 12 is an example of flow chart describing a procedure for searching a multidimensional indexing structure to retrieve the database item having the highest score according to the scoring function obtained using the procedures of Figure 9 or Figure 10.

Figure 13 is a flow chart depicting an example of how to decide whether a node of the multidimensional indexing structure is a candidate during the search.

Figure 14 is a simplified flow chart depicting another procedure for searching a multidimensional indexing structure to retrieve the database item having the highest score according to the scoring function obtained using the procedures of Figure 9 or Figure 10.

Figure 15 is a simplified flow chart depicting a further procedure for searching a multidimensional indexing structure to retrieve the database item having the highest score according to the scoring function obtained using the procedures of Figure 9 or Figure 10.

Detailed Description

Definitions

The following definitions will be used in the description of the preferred embodiments of the current invention:

Feature Space

By feature space we denote the domain of definition of the set of attributes used to search the database. In a preferred embodiment, a traditional database is searched using the values in a subset of the existing numerical columns, the feature space is a multidimensional Euclidean space where each dimension is associated with a different column.

More generally, if a column is numeric, then the feature space is the Cartesian product of the real line and of the feature space of the remaining columns. If a column is categorical or it can take a finite number of values, the feature space is the Cartesian product of the set of possible values of that column and of the feature space of the remaining columns. Hence, a database column corresponds to a unique dimension in the feature space.

In another preferred embodiment, an object-relational database is searched, using the values of a group of attributes. The feature space is the domain of definition of the set of attributes used in the search. If an attribute is numeric, then the feature space is the Cartesian product of the real line and of the feature space of the set of remaining attributes. If an attribute is categorical, or takes a finite number of values, then the feature space is the Cartesian product of the set of possible values of the attribute and of the feature space of the remaining attributes. Hence, each attribute corresponds to a unique dimension in the feature space.

Search Space

The collection of database record considered during the current step of the search. All the embodiment are described in reference to a database, however one skilled in the art would appreciate that the same descriptions apply to different assumptions: for example instead of a database the invention can be applied to a collection of databases, a federation of databases, a spreadsheet, to data provided manually by an operator, to data automatically produced by measurement equipment, to data provided by a knowledge provider, including, but not limited to, real-time stock prices etc.

Example

The present invention is concerned with queries specified by providing examples. An example is any element of the feature space. In practice, the user can provide, as an example, a database record. In this case, only the columns (or attributes) of the database record that correspond to dimensions in the feature space are considered, and the others are discarded. In a preferred embodiment, the database contains multimedia objects, and the feature space is obtained from low-level numerical or categorical features, such as texture descriptors in the case of images. The user provides as an example a multimedia object, such as an image, or a portion of it. The low-level features are extracted from the multimedia object. In this embodiment, the term example refers interchangeably to the multimedia object and to its representation in terms of low-level features.

Example set

An example set in the spirit of the current invention is a collection of examples that the user deems as similar with respect to some characteristics. The user need not specify exactly which characteristics are homogeneous across the examples in the example set. In the spirit of the current invention, an example set always contains examples that are similar with respect to one

or more characteristics, which is interpreted as a defining characteristics of the example set. The terms "sample" and "example" are synonyms in the spirit of the current invention.

Scoring function

A scoring function is a function defined on the feature space, that assigns a numeric value, called score, to each element of the feature space. In the present invention scoring functions are used to decide which elements of the database must be retrieved in response to a given query. The **scoring function properties** that are relevant to the present invention are the following:

a. A higher value of the scoring function is interpreted as a better fit to the query.
b. Without loss of generality, in the spirit of the present invention a scoring function takes values between zero and one.

It will be immediately apparent to one skilled in the art that a scoring function taking values in any other range can be transformed to a scoring function that takes values between zero and one using a continuous, monotonic transformation which maintains the desired property. Additionally, in the spirit of the present invention, a score of 1 is interpreted as a perfect fit, while a score of 0 as a complete misfit to the query.

Figure 1 depicts an example of a system having features of the present invention. A search engine (101) is communicates to a user interface (103) either directly, or through a local area network, or through a wide area network (102) such as the Internet. The user interface (103) allows the user to construct queries using multiple positive and negative example sets, to submit the query to the search engine (101), to visualize the results produced by the search engine, and to iteratively refine the search. The search engine interfaces directly, via a local area network, or through a wide area network (104) with relational databases (105), object-relational databases (106), or multimedia repositories (107), which manage data stored on disk or in a storage server (108). In the spirit of the present invention, the search engine can also be integrated with the relational databases, the object-relational databases and the multimedia repositories, and in this case it can be thought as the server-side component that manages the communication with the user interface (103).

A user provided example set is in a one-to-one correspondence with a set of points in the feature space. **Figure 2** shows a set of such points (201) in a two-dimensional feature space, having two orthogonal axes (202) and (203) each corresponding to a different feature or attribute.

Figures 3 to 12 show a preferred embodiment of how multiple positive and negative example sets are used to construct on-line a scoring function that can be used in conjunction with an existing multidimensional index to execute the user-submitted query.

Figure 3 shows how the example set is used by the search engine to infer what aspects (features or attributes) are perceived as homogeneous by the user, and to construct a scoring function. First, the system infers, from the samples in the example set, a "characteristic example" (301). In a preferred embodiment, where the attributes are numeric, the characteristic example is the centroid of the example set, namely, the value of each of its attributes is the arithmetic average of the corresponding attribute values computed over the examples in the set. In another preferred

embodiment, where the attributes are numeric, the characteristic example has attribute values equal to the median of the corresponding attribute values computed over the samples in the set. In another embodiment, where some attributes are categorical, the value of each categorical attribute of the characteristic example are the mode (namely, the most frequent value) of the corresponding attribute values computed over the samples in the set. Second, the system infers, from the samples in the example set, a description of the dispersion of the samples in the example set around the characteristic example. In a preferred embodiment, the description of the dispersion is the covariance matrix of the examples. In another preferred embodiment, where the interaction between different attributes is ignored, the dispersion is captured by the standard deviation of the individual attributes of the examples around the characteristic example: the standard deviation of the ith attribute is

$$s[i] = \{[(\mathbf{x}_1[i]-\mathbf{c}[i])^2+(\mathbf{x}_2[i]-\mathbf{c}[i])^2+\ldots (\mathbf{x}_N[i]-\mathbf{c}[i])^2] / N \}^{1/2} \qquad (2)$$

where $\mathbf{x}_j[i]$ is the value of the ith attribute for the jth example, $\mathbf{c}[i]$ is the value of the ith attribute of the characteristic example, and N is the number of samples in the example set. The standard deviations are indicated in Figure 3 by bars with arrows (302 and 303), aligned with the coordinate axes: the longer the bars, the bigger the dispersion around the characteristic example. Using the standard deviations alone is preferable to using the full covariance matrix when the number of samples in each example set is smaller than the square of the number of attributes on which the search is performed; this is in general a common situation. It is apparent to one skilled in the art that other measures of dispersion can be used in the spirit of the current invention: different embodiments of the current invention use central moments rather than standard deviation; another embodiment uses an order statistics of the differences $| | \mathbf{x}_1[i]-\mathbf{c}[i] |, | \mathbf{x}_2[i]-\mathbf{c}[i] |, \ldots, | \mathbf{x}_N[i]-\mathbf{c}[i] |$ for each attribute i, where the notation $| x |$ denotes the absolute value of x. In the spirit of the current invention, further characterizations of the dispersion of the samples in the example set around the characteristic example can be computed, for example third moments capturing asymmetry.

Figure 4 shows an example of the flow chart for constructing a scoring function from a single example set (401). The characteristic example (403) is computed (402) for instance using one of the methods taught in the description of Figure 3. The dispersion of the data (405) around the characteristic example is computed in step (404), for instance using one of the methods taught in the description of Figure 3. The characteristic example (403) and the characterization of the dispersion (405) are then used as input to Step (406), which produces a scoring function, namely a function that assigns a value, to each possible database item, indicating how well the database item satisfies the query. Figure 6 will describe a preferred embodiment for step (406). In the spirit of the current invention, a similarity or dissimilarity function other than a scoring function taking values between zero and one can be used, as previously discussed. In the spirit of the current invention, the scoring function will have the following two **Required Properties**, described with reference to Figure 5:

1. Consider two database items (or rows) (502 and 503) whose corresponding attribute values are all identical, except the values of one specific attribute, say, the ith. The database item whose ith attribute value is closer to the ith attribute value of the characteristic example (501) will have a higher score than the other database item, namely, it will match the query better

than the other database item. In Figure 5, database example (503) will have a higher score than database example (502). Hence, if two database items are identical except for the value of one attribute, the database item whose attribute value is more similar to the corresponding attribute value of the characteristic example satisfies the query better.

2. Consider two database items, say **x** (504) and **y** (505), whose corresponding attributes values are identical, except two, say the ith and the jth. Assume additionally that $| \mathbf{x}[i] - \mathbf{c}[i] | = | \mathbf{y}[j] - \mathbf{c}[j] |$ and that $| \mathbf{x}[j] - \mathbf{c}[j] | = | \mathbf{y}[i] - \mathbf{c}[i] |$. In Figure 5 the differences $| \mathbf{x}[i] - \mathbf{c}[i] |$, $| \mathbf{y}[i] - \mathbf{c}[i] |$, $| \mathbf{x}[j] - \mathbf{c}[j] |$ and $| \mathbf{y}[j] - \mathbf{c}[j] |$ are graphically indicated by the segments (506), (507), (508) and (509) respectively, and the characteristic example **c** is (501). Let, as in the figure, $| \mathbf{x}[i] - \mathbf{c}[i] | > | \mathbf{x}[j] - \mathbf{c}[j] |$. Let the dispersion of the example set around the characteristic example be larger along the ith attribute than along the jth attribute. Then the database item **x** has higher score than database item **y**. Hence, dissimilarity from the characteristic example along attributes with larger dispersion of the example set is less important than dissimilarity along attributes with smaller dispersion of the example set. We can interpret this requirement as follows: if the example set has small dispersion along a specific attribute, that attribute captures well the homogeneity of the samples in the example set; if an attribute has large dispersion, it does not capture well homogeneity. Attributes capturing homogeneity well are more important that attributes that poorly capture homogeneity: hence, dissimilarity from the characteristic example along attributes capturing homogeneity well is weighted more than dissimilarity capturing homogeneity poorly.

Figure 6 shows a flow chart of a preferred embodiment for the computation of the score of a generic database example (601), so as to satisfy the above constraints. The characterization of the dispersion (602) and the characteristic example (603) are used by step (604). Step (604) computes the weighted Euclidean distance (605) between the characteristic example (603) and the database item (601) using the dispersion characterization (602) to compute the weights. In a preferred embodiment, where the dispersion characterization consists of the standard deviations of the individual attributes, the weights used in the weighted Euclidean distance are the reciprocal of said standard deviations. One skilled in the art would appreciate that the distance (605) is an indicator of dissimilarity that has the two required properties. In other embodiments, weighted Minkowsky distances are used instead of the Euclidean distance (605). One skilled in the art would appreciate that other distances as well as semi-metrics can be used in step (605). In fact, the property of symmetry is not required to satisfy requirements 1. and 2., and, for example functions satisfying non-negativity and the triangular inequality can be used instead of a distance in step (605).

Step (606) converts the distance (605) into a score (607), in such a way that the required scoring function properties (a) and (b). In one embodiment, step (606) produces $\mathbf{exp}(-\mathbf{d})$, where **exp** is the exponential function and **d** is the weighted Euclidean distance (605). In another embodiment, step (606) computes $k / [\, 1 + \mathbf{exp}(a - b * \mathbf{d}) \,]$, where k, a, and b are positive constants, and k is equal to $(1 + \mathbf{exp}(a))$ in order to satisfy property (b) of the scoring function. One skilled in the art would appreciate that different functions can be used in step (606) in the spirit of the present invention.

In an embodiment of the present invention, the scoring function satisfies the following requirements:

i. Database items that are very similar to the characteristic example have a score equal to 1: it is likely that the user cannot provide an ideal example set, hence the query is approximate; database items that are very similar to the characteristic example are declared perfect matches. The user can further refine the query as described later, if desired.
ii. Database items that are very dissimilar to the characteristic example have a score of zero: it is not important to retrieve database items that match the query very poorly.

In a preferred embodiment, step (606) satisfies properties (a), (b), (i) and (ii) by using two thresholds, T_1 and T_2. If the distance (605) is below T_1, the score is 1, if the distance (606) is above T_2, the score is 0, and if it is between T_1 and T_2, the score is computed as a monotonically decreasing function of the distance that satisfies properties (a) and (b).

Figure 7 shows two such possible functions. Function (701) is piecewise linear and continuous. Function (702) has a squared root behavior between the thresholds T_1 and T_2. Hence, computing the score with function (702) and the Euclidean distance (605) is equivalent to computing the score using the squared Euclidean distance instead of the Euclidean distance (605) and function (701), after appropriately adjusting the thresholds. The scoring function assigns a score to every point of the feature space. **Figure 8** graphically depicts an example of scoring function constructed with the example set of Figures 2 and 3, using the procedure illustrated in Figures 4, 6 and 7. The examples (201), the characteristic sample (301) and the dispersion indicators (302) and (303) are shown as reference. The lines (801) connect the set of points of the feature space having the same score. In a D-dimensional feature space, the lines (801) are (D-1)-dimensional surfaces. Only some of the lines are drawn, and the corresponding scores (802) are shown. Note that the scores of more external lines are lower than the scores of internal lines.

These numbers are provided only for illustration purposes, and they would vary depending on the actual detail of the embodiment. Similarly, the lines (801) depicted in the figure are ellipses, since in this embodiment the distance (605) is a weighted Euclidean distance, but in other embodiment can have very different forms: for instance, if in an embodiment the weighted Manhattan distance is used in step (605), the lines (801) would be romboids.

Figure 9 illustrates an embodiment of the method for using multiple example sets. The user provides multiple example sets (901) to the system. The system analyzes one example set at a time: the iterator (902) graphically indicates the sequential iteration. Step (903) takes a different example set at each iteration, and constructs the corresponding scoring function, for example as illustrated in Figures 4, 5 and 7. When the iterator (902) terminates because all the example sets (901) have been analyzed, step (903) has produced a collection of individual scoring functions (904), one per each of the multiple example sets (901). Step (905) combines the scoring functions of the collection (904) to produce the overall scoring function (906).

In a preferred embodiment, the combining step (905) satisfies the following requirements:

I. Points in the feature space that have low score with respect to all the scoring functions of the positive example sets have low score.
II. Points in the feature space that have high score with respect to all the scoring function of the negative example sets have low score.
III. Points in the feature space that have high score with respect to at least one scoring function of the positive example sets and low score with respect to all the scoring functions of the negative example sets have high score.

The three requirements imply that database points that look like negative examples will have low score, database points that look like positive examples but not like negative examples will have high score, and database points that are very different from any positive example have low score.

In a preferred embodiment, Step (905) combines the collection of individual scoring functions (904) as follows: let $f_1(.),...,f_p(.)$ be the scoring functions of the positive example sets, where p is the number of positive example sets, and let $g_1(.),..., g_n(.)$ be the scoring functions of the negative example sets, where n is the number of negative examples; Step (905) produces a function $S(.)$ that assigns to each element **x** of the feature space the score

$$S(\mathbf{x}) = \min \{ \max_{i=1...p} \{ f_i(\mathbf{x}) \} , 1 - \max_{j=1...n} \{ g_j(\mathbf{x}) \} \} \qquad (3)$$

Namely, step (905) computes the maximum score with respect to all the positive example sets, subtracts from 1 the maximum score with respect to the negative example sets and computes the minimum. Equation (3) satisfies I., II., and III. If a point in the feature space has low score with respect to all positive example sets, then $\max_{i=1...p} \{ f_i(\mathbf{x}) \}$ is a small number, and $S(\mathbf{x})$ is small irrespective of the value of $1 - \max_{j=1...n} \{ g_j(\mathbf{x}) \}$, hence Equation (3) satisfies I. If a point **x** has high score with respect to at least one of the negative example sets, $\max_{j=1...n} \{ g_j(\mathbf{x}) \}$ is a large number, hence $1 - \max_{j=1...n} \{ g_j(\mathbf{x}) \}$ is a small number and $S(\mathbf{x})$ satisfies II. It is immediate to see that III. is also satisfied. In the spirit of the current invention, other methods for combining individual scores can be used.

The multiple example sets can also be used together, rather than sequentially, to produce a scoring function. We assume that no representative sample of positive example sets is identical to the representative sample of negative example sets: if this happens, in a preferred embodiment the representative sample of the positive example set is ignored. In a preferred embodiment, depicted in **Figure 10**, Step (1002) iterates over the multiple example sets, and Step (1003) computes the representative function and dispersion characterization for each of the example sets. The result collection (1004) is then jointly analyzed in Step (1005), which selects appropriate scoring functions. In a preferred embodiment, Step (1005) modifies the dispersion characterizations so that each negative example gives very low score to all positive examples, where the score is computed as previously described in the present invention, and each positive example gives very low score to all negative example. In another preferred embodiment, where Step (606) of Figure 6 satisfies requirements i. and ii., Step (1005) modifies the values of the thresholds T_1 and T_2. In another preferred embodiment, Step (1005) selects the dissimilarity function among a collection of dissimilarity function that best satisfies I. II. and III. In yet another embodiment, Step (1005) selects from a collection the dissimilarity-to-score conversion function that best satisfies I. II. and III. It is clear to one skilled in the art that strategies that combine two or more of those used in the listed preferred embodiments can be used in the spirit of the present invention: for example, Step 1005 can simultaneously modify the dispersion characterizations and the values of the thresholds T_1 and T_2. Step (1005) yields a collection of modified scoring functions (1006): $f'_1(.),...,f'_p(.)$ for the positive example sets and $g'_1(.),...,$ $g'_n(.)$ for the negative examples. Step (1007) combines the scoring functions. In a preferred embodiment, Step (1007) uses Equation (3) with f' instead of f and g' instead of g.

Figure 11 shows an example of a scoring function that can be obtained with three positive and two negative example sets. Comparison with Figure 8 reveals the flexibility of the method for constructing scoring functions from multiple positive and negative example sets.

It would be apparent to one skilled in the art that the procedures described in the figures can also be adapted to the case where the user adds information to the example sets in the form of a score or a qualitative assessment (i.e., by telling that a certain example set is 'very positive', while another is 'moderately negative'). For example, the maximum score of each example set would be changed from 1 to the score provided by the user or to a value corresponding to the qualitative assessment. Additionally, the mutual influence of different example sets can also be dependent on the scores or qualitative assessments of the example sets, by adding appropriate requirements to I., II., and III., such as

IV. The influence of the example sets on points depend on the score of the example sets: example sets with more extreme scores, either positive or negative, have higher influence than example sets with less extreme scores.

Searching a database by computing the score of each of the contained items using, for instance, Equation (3) can be a computationally intensive and time-consuming operation. The value of the present invention also lies on the ability of performing the search interactively, allowing the user to evaluate the results of the search, and modify the query by iteratively adding new example sets or removing existing example sets from the query specification. This interactive query refinement process is possible only if the query engine quickly returns results. It is therefore necessary to use an indexing structure that supports searches based on the scoring functions taught in the present invention. Also, since indexing structures are computationally expensive to construct and to maintain (construction is a rare occasion, while maintenance is required every time a new item is added to the database or an existing item is removed), a method for searching existing indexing structures is preferable to a method that requires a new indexing structure. One skilled in the art would appreciate that the method for searching an existing indexing structure can also be applied to a new indexing structure.

Numerous multidimensional indexing structures have been proposed in the literature. The vast majority (see [GaeGunt98]) are related to the R-Tree, the Grid File, the K-D Tree and the Quadtree. These methods perform recursive partitioning of the search space using hyperplanes (or hyperrectangles) aligned with the coordinate axes (that represent the different attributes of the database items). The embodiments described in this patent refer to a specific multidimensional indexing structure called "Ordered Partition", taught in [KimPark96]. The only property of this indexing structure that is relevant to the invention is the fact that the search space is partitioned using hyperplanes aligned with the coordinate axes, namely the general property of the indexing structures mentioned above. It has been chosen to describe the embodiments because it is simple to describe. Given a database where items have D dimensions, the ordered partition divides the search space into n parts using n-1 (D-1)-dimensional hyperplanes that are perpendicular to the first dimension. These hyperplanes are placed in appropriate positions in order to have approximately the same number of database items in each of the n parts. Then each of the n parts is separately divided into n parts using n-1 hyperplanes that are perpendicular to the second dimension, in such a way that each of the new parts contains approximately the same number of database items. The process is recursively repeated, as taught in [KimPark96], until all dimensions are exhausted, or the individual cells produced by the recursive partitioning contain

fewer than a predefined number of points. The choice of n is performed to satisfy these criteria, as taught in [KimPark96]. The recursive partition can be represented (graphically and as a data structure within a computer) as a tree: each node corresponds to a cell and its descendants correspond to all the cells that are obtained by partitioning with the hyperplanes along the next dimension. The leaves, namely the nodes that do not have descendants, contain the actual database items or a reference to them. The recursive partition can be efficiently used for nearest neighbor queries as taught in [KimPark96] because of its tree structure.

In all the embodiments described below, the following assumptions are made: properties a., b., 1., 2., I., II. and III. are satisfied.

Figure 12 shows a flow chart of a preferred embodiment to search a recursive partitioning indexing structure. The user provides multiple positive and negative example sets (1201) and the number of desired results (1202). The search algorithm returns the number of desired results. Every database item which is not returned by the search has score smaller than the score of any of the returned results. Step (1203) computes the scoring functions (1204), for instance as previously described in the current invention. Step (1205) initializes the search by setting the current node to the root of the search tree and by initializing the list of current results (1206) to an empty list. The query execution then proceeds with Step (1207) which determines if the current node is a leaf. In a preferred embodiment Step (1207) relies on mechanisms of the existing indexing structure to determine if the current node is a leaf. If the current node is a leaf, Step (1208) searches the leaf. In a preferred embodiment, Step (1208) consists of searching the leaf node exhaustively, namely, it computes the score of each of the database items that are associated with the leaf node using the overall scoring function (1204), compares it with the scores in the current list of results (1206), and adds the item with its score to the list (1206) if the list (1206) contains less than the desired number of results (1202), or if the score of at least one of the items contained in the list (1206) is lower than the score of the database item being analyzed. When all the items associated with the leaf are analyzed and the list (1206) is appropriately updated, step (1209) sets the current node to the parent of the current node. Step (1210) checks if the current node is the root. If the current node is the root, Step (1211) checks whether there are unsearched candidate children. A child is a candidate if it can contain database items with score higher than the lowest score of any database item in the result list (1206), or if the result list (1206) contains less than the number of desired results (1202). If the root does not have candidate children, the computation terminates (1212) and the current result list (1206) is the result of the query. If Step (1210) determines that the current node is not the root, if Step (1211) determines that there are candidate children, or if Step (1207) determines that the current node is not a leaf, the computation continues from Step (1213). Step (1213) determines if there are unsearched children. If there are not unsearched children, Step (1209) is executed as described above. If there are unsearched children, Step (1214) determines if there are candidates among the unsearched children. If there are no candidates, Step (1209) is executed as described above, otherwise Step (1215) sets the current node to the best candidate child and the computation continues from Step (1207).
Steps (1214), (1215) and (1211) use the scoring functions (1204) and the result list (1206). One skilled in the art would recognize that Figure 12 describes a branch-and-bound search algorithm. The novelty of the algorithm taught in this invention is the use of multiple positive and negative

example sets in determining the existence of candidate children and in determining the best candidate child.

Figure 13 shows the flow chart of a preferred embodiment of the method for determining whether a node is a candidate and to compute its score, when Equation (3) is used to compute the scores. Step (1303) initializes the positive node score (the score with respect to the positive example sets) to 0 and the negative node score (the score with respect to the negative example sets) to 1 (1304). The iterator (1305) considers each negative example set in sequence. If there are unused negative example sets, Step (1306) uses the corresponding scoring function *g()* (1301) and determines the minimum possible score of any point in the currently analyzed node, according to the scoring function *g()*. Call this minimum possible score *s*. Step (1307) compares (1 - *s*) with the minimum of the scores in the current result list. If the list is full (i.e., it contains the number of desired results) and (1-*s*) is smaller than the minimum score in the list, Step (1308) declares that the node is not a candidate, and the procedure ends. Otherwise, step (1309) updates the negative node score (1304): if the current negative node score is smaller than (1-*s*), the negative node score (1304) is unchanged, otherwise it is set to (1-*s*), and the next negative example set is used. When the last negative example set has been used, iterator (1301) uses each positive example set in turn. For each positive example set, step (1311) uses the corresponding scoring function *f()* and determines the maximum possible score of any point within the portion of the search space corresponding to the current node. Call this quantity *r*. Step (1312) updates the positive node score (1304) by comparing it to *r*. If *r* is larger than the current positive score, the positive score is set to *r*. When the last positive example set has been used, Step (1313) computes the node score as the minimum of the positive node score and of the negative node score. Step (1313) then compares the node score with the lowest score in the result list (1302). If the node score is smaller, step (1308) declares that the node is not a candidate. Otherwise, Step (1314) declares the node to be a candidate.

In another preferred embodiment, illustrated in **Figure 14**, the user provides multiple positive and negative example sets (1401) and the number of desired results (1402) when submitting the query. In response to the query, the system computes individual and overall scoring functions (1404) n step (1403), initializes the search in Step (1405) by setting the current node to the root, and initializing the result list (1406) to empty, then uses the negative examples only to assign a score to each node of the tree in step (1407), and then searches the tree with the positive examples alone in Step (1408), which also updates the result list (1406). Step (1408) also uses the assigned negative scores to decide whether nodes are candidates. Step (1409) returns the result list (1408) to the user. In a preferred embodiment, Step (1407) is performed by means of Steps (1305), (1306) and (1309) as described in Figure 13. In a preferred embodiment, Step (1408) is performed by means of Steps (1310), (1311) and (1312) as described in Figure 13.

In another preferred embodiment, described in **Figure 15**, the user provides multiple positive and negative example sets (1501) and the number of desired results (1502) when submitting the query. In response to the query, the system computes individual and overall scoring functions (1504) n step (1503), initializes the search in Step (1505) by setting the current node to the root, and initializing the result list (1506) to empty. Iterator (1507) iterates over the individual positive example sets. For each positive example set, Step (1508) descends the tree by identifying candidate nodes until it reaches candidate leaves using the scoring functions of the

specific positive example set and of all the negative example sets. Step (1509) uses the scoring function all positive and negative example sets. Steps (1508) and (1509) are continued until no more candidate leaves exist. During the descent in Step (1508), scores are associated to the nodes of the tree. If the node does not have a score, its positive and negative scores become the ones computed in step (1508). If the node already has a score, the negative score is not changed (since it has already been computed using all negative examples), and the positive node score becomes the maximum of the current positive node score and the positive node score computed by step (1508). The overall node score is then the minimum between the positive and the negative node scores. If a node already has scores, in a preferred embodiment Step (1508) does not recompute its negative node score. In a preferred embodiment, if the negative node score is smaller than the minimum of the scores in the result list (1506), the node is automatically declared a non-candidate node by step (1508). Step (1509) also marks the searched leaf nodes. Since the search is performed with all positive and negative scoring functions, it is unnecessary to search the same leaf node twice. In another preferred embodiment, Step (1509) uses only one positive scoring function, namely, the same used by Step (1508), and all negative scoring functions.

When all the positive examples have been exhausted, Step (1510) returns the result set (1506).

It would be clear to one skilled in the art that the methods for searching can be easily adapted to multidimensional indexing structures where the hyperplanes or hyperrectangles are not necessarily parallel to the coordinate axes, **and** to multidimensional indexing structures where partitioning is performed using other types of surfaces, including spheres and polygons.

It would also be clear to one skilled in the art that the methods for searching can be trivially adapted to multidimensional indexing structures that store database items or references to database items both at leaves and at internal nodes.

Examples of Claims

Claim 1: in a computer system containing a database supporting multidimensional indexing, a method for retrieving from the database the most relevant items a query specified by the user via a multiplicity of positive and negative example sets, comprising the steps of

- Constructing a scoring function from the multiplicity of positive and negative example sets, said scoring function giving higher scores to database items that are more closely relevant to said query, said scoring function being suitable for using with a multidimensional indexing structure.
- Retrieving via said multimensional indexing structure the items of said database that have the highest score as computed using said scoring function.

Claim 2: the method of claim 1, where said scoring function is obtained by combining the scoring functions of said multiplicity of positive example sets and negative example sets.

Claim 3: the method of claim 2, where said scoring function assigns to each element of the search space a score equal to the minimum of the maximum of the scores assigned to said element by the scoring functions of said positive example sets, and the minimum of 1 minus the scores assigned to said element by the scoring functions of said negative example sets.

Claim 4: the method of claim 1 (2, 3), where the scoring function of each of said multiple example sets is obtained by converting with a distance-to-score conversion function a semi-metric obtained using said each of said multiple example sets.

Claim 5: the method of claim 4, where the semi-metric is a weighted Minkowsky distance from a representative sample of the examples in said each of said multiple example sets, where the weights are calculated using the examples in said each of said multiple example sets.

Claim 6: the method of claim 5 where the weights are the inverse of the standard deviations of the examples in said each of said multiple example sets.

Claim 7: the method of claim 5 (, 6) where the representative sample is the centroid of the examples in said example set.

Claim 8: the method of claim 4 (5, 6, 7) where the distance-to-score conversion function is a monotonically non-increasing continuous function having value 1 at the origin and zero at infinity.

Claim 9: the method of claim 8 where the monotonically non-increasing function is equal to 1 between zero and T1, and to zero after T2.

Claim 10: the method of claim 2 (3,4,5,6,7,8,9) where combining the scoring functions of the positive example sets and of the negative example set comprises the steps of

- Modifying the scoring functions of said multiplicity of positive example sets and negative example sets so that the scoring functions of the positive example sets assign low score to the representative samples of the negative example sets, and so that the scoring functions of the

negative example sets assign low score to the representative samples of the positive example sets.
- Combining the modified scoring functions of said multiplicity of positive example sets and negative example sets.

Claim 11: the method of claim 1 where the user specifies the number of items to retrieve from the database.

ETC: TO COVER THE OTHER PREFERRED EMBODIMENTS DESCRIBED ABOVE

Claim 12: the method of Claim (1,2,3, etc.) Additionally comprising the step of
- Searching a multidimensional indexing structure to retrieve from the database the items having the highest score.

Claim 13: the method of claim 12 where the multidimensional indexing structure is used to execute different queries.

Claim 14: the method of claim 12 (13) where the multidimensional indexing structure is based on the recursive partition of the search space using hyperplanes parallel to the coordinate axes.

Claim 15: the method of claim 13 where searching the multidimensional indexing structure comprises the steps of
- Using the scoring functions of said multiplicity of positive and negative example sets to search the tree, by identifying candidate nodes
- Using the scoring functions of said multiplicity of positive and negative example sets to exhaustively score the items stored at the leaves of said multidimensional indexing structure.

Claim 16: the method of claim 15 where identifying candidate nodes comprises the steps of
- Computing for each scoring function of said positive example sets, the maximum possible score of an item stored at the node or at one of the descendants of the node.
- Computing the maximum of said maximum scores
- Computing for each scoring function of said negative example sets, the minimum possible score of an item stored at the node or at one of the descendants of the node.
- Computing the minimum of one minus said minimum scores.
- Computing the minimum of said maximum of said maximum scores and of said minimum of one minus said minimum scores
- Comparing said computed minimum to the minimum of the scores in the current result set.
- Declaring that a node is a candidate if said minimum is not smaller than the minimum of the scores in the current result set.
- Declaring that the node is not a candidate otherwise.

Claim 17: the method of claim 15 (, 16) where the search is performed by using said scoring functions of said positive example sets one at a time in conjunction with all scoring functions of said negative example sets.

Figure 1

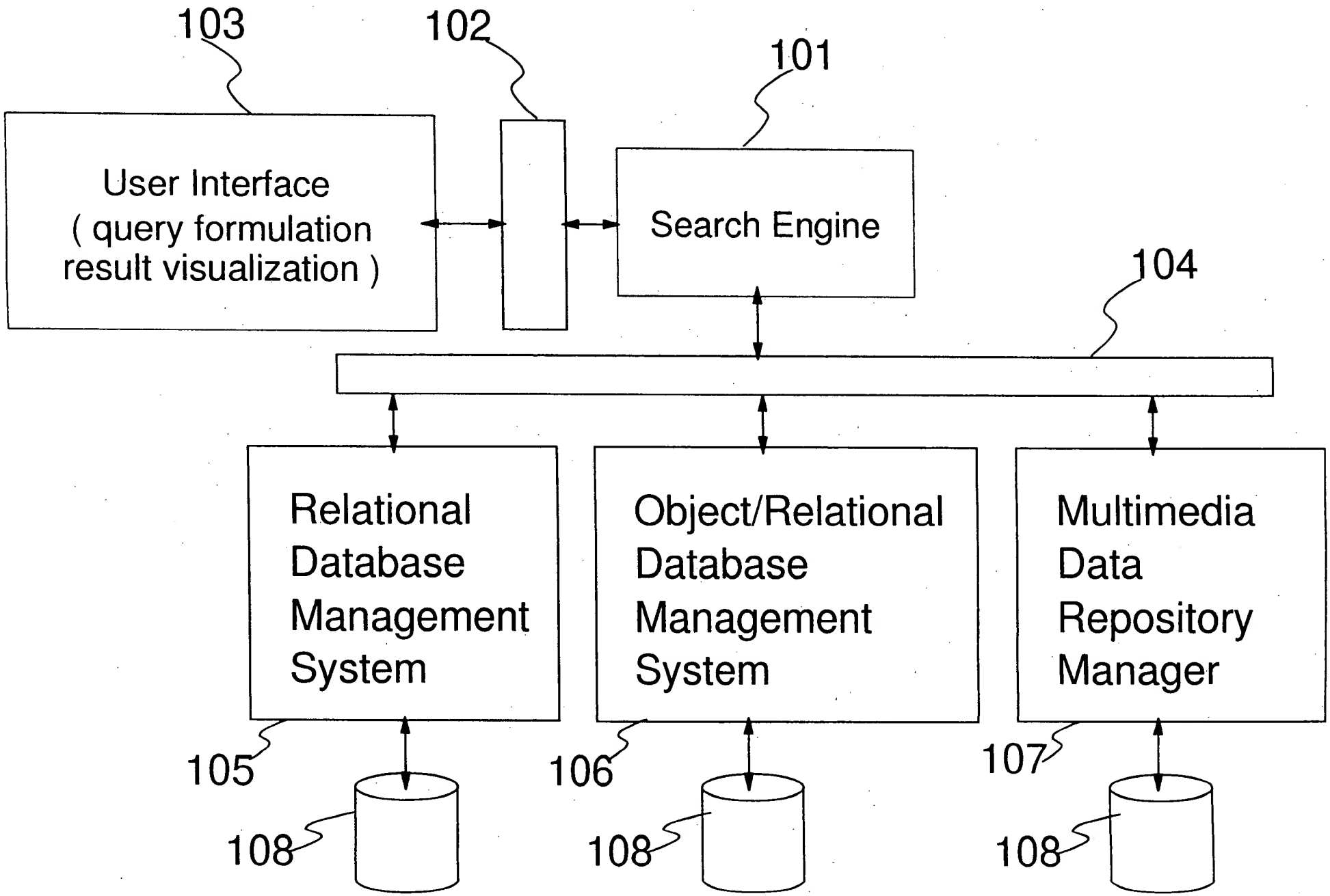

103
102
101
User Interface
(query formulation
result visualization)
Search Engine
104
Relational
Database
Management
System
Object/Relational
Database
Management
System
Multimedia
Data
Repository
Manager
105
106
107
108
108
108

Figure 2

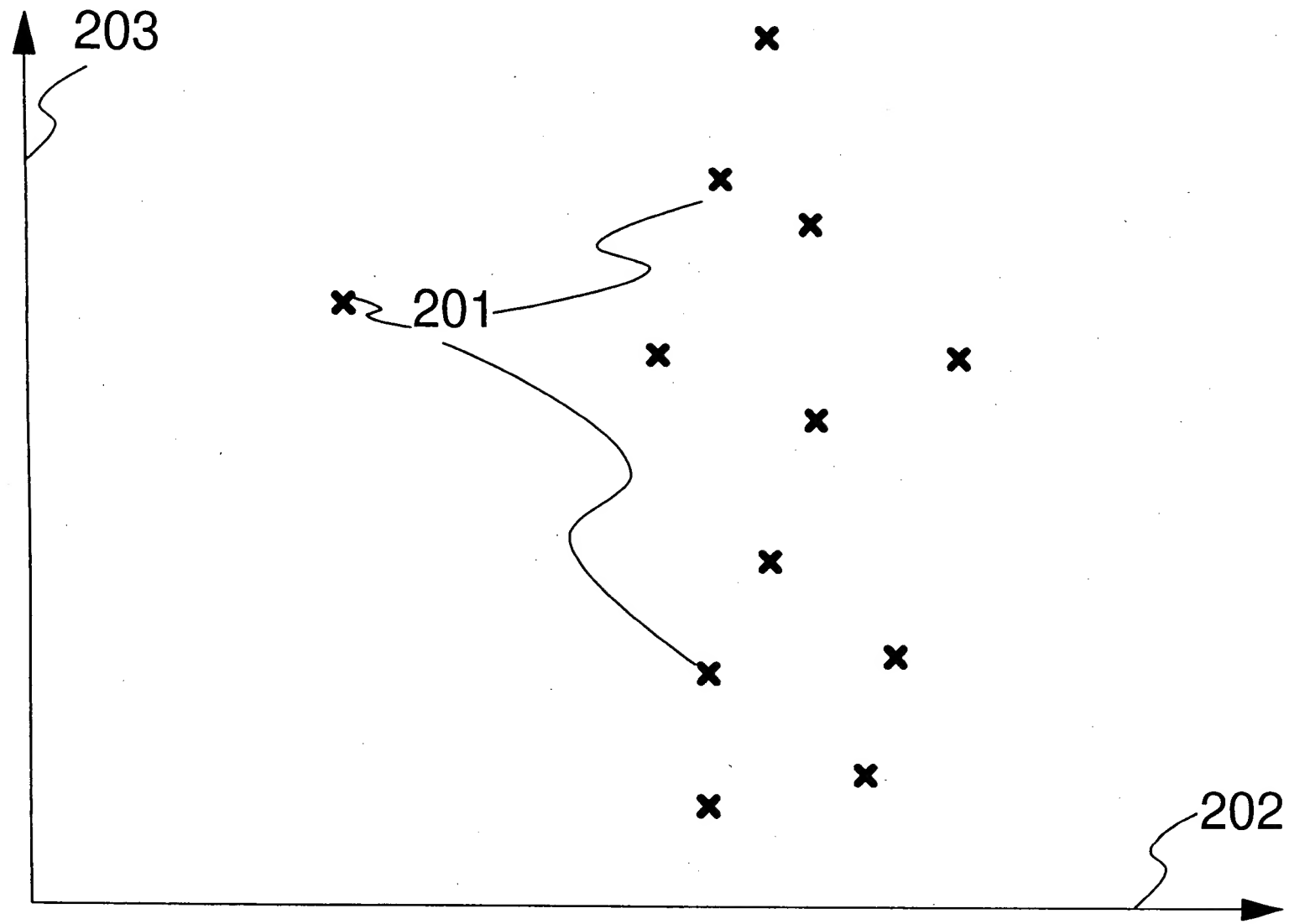


Figure 3

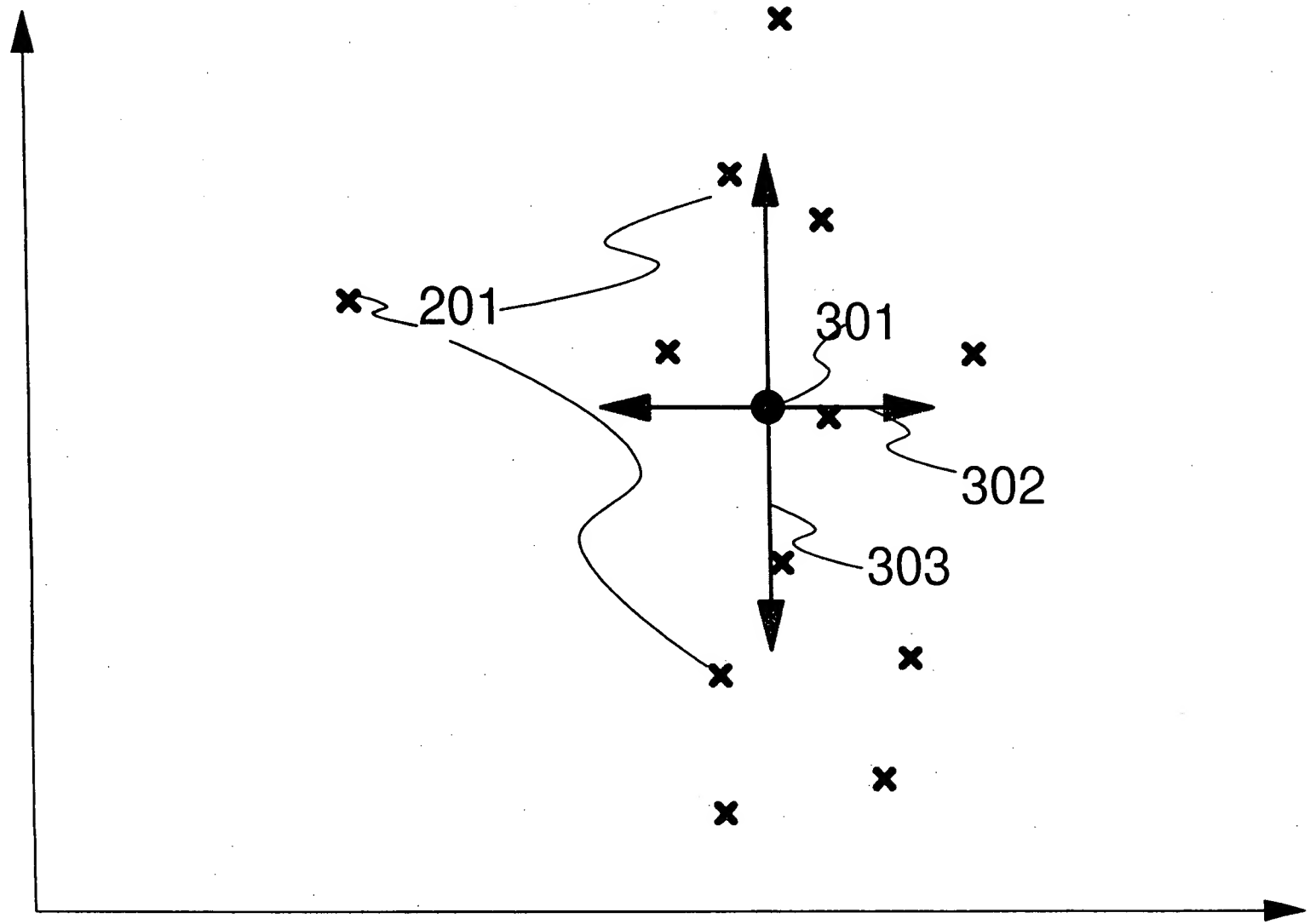


Figure 4

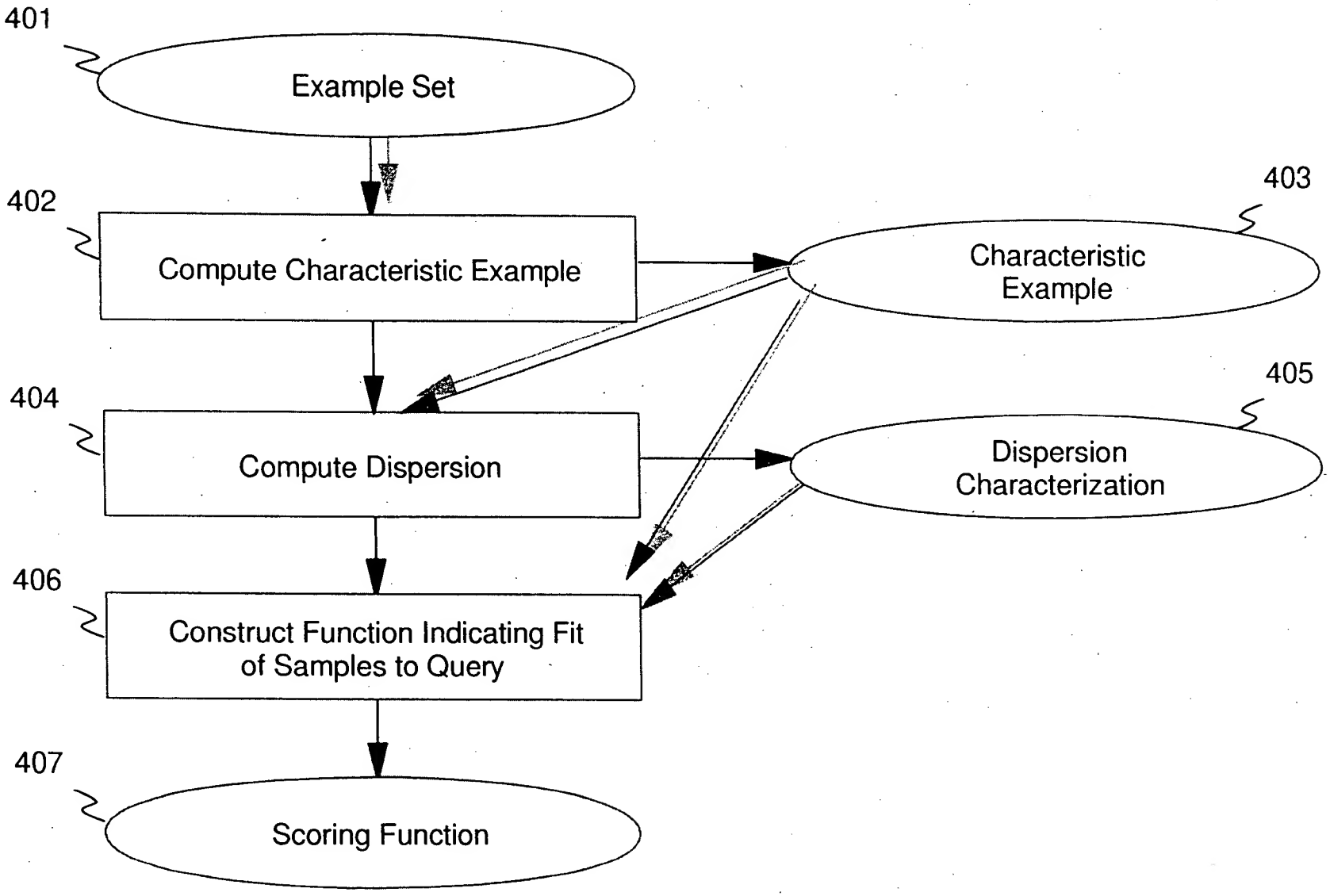

401
Example Set
402
Compute Characteristic Example
403
Characteristic Example
404
Compute Dispersion
405
Dispersion Characterization
406
Construct Function Indicating Fit of Samples to Query
407
Scoring Function

Figure 5

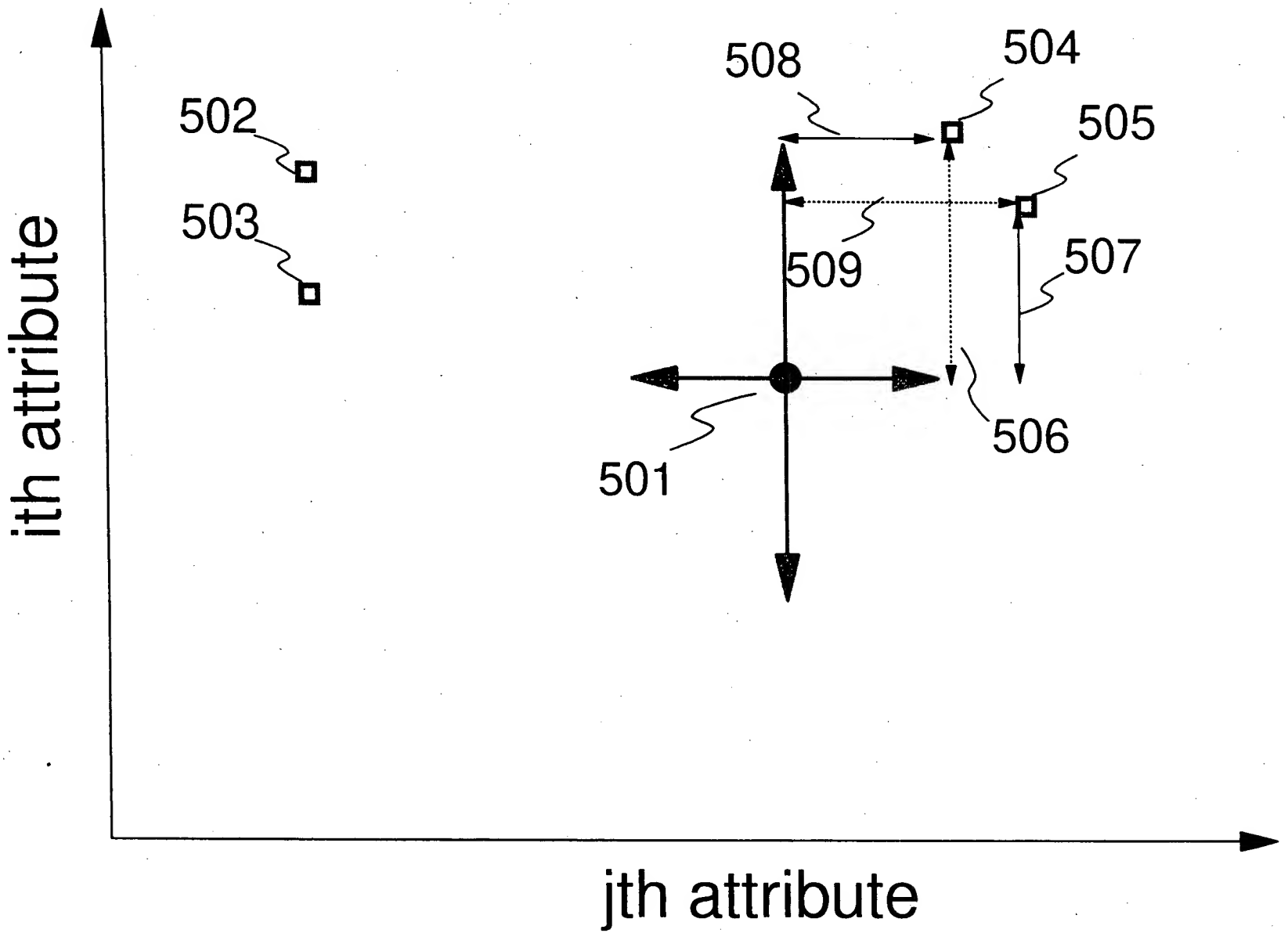

ith attribute
jth attribute
502
503
508
504
505
509
507
501
506

Figure 6

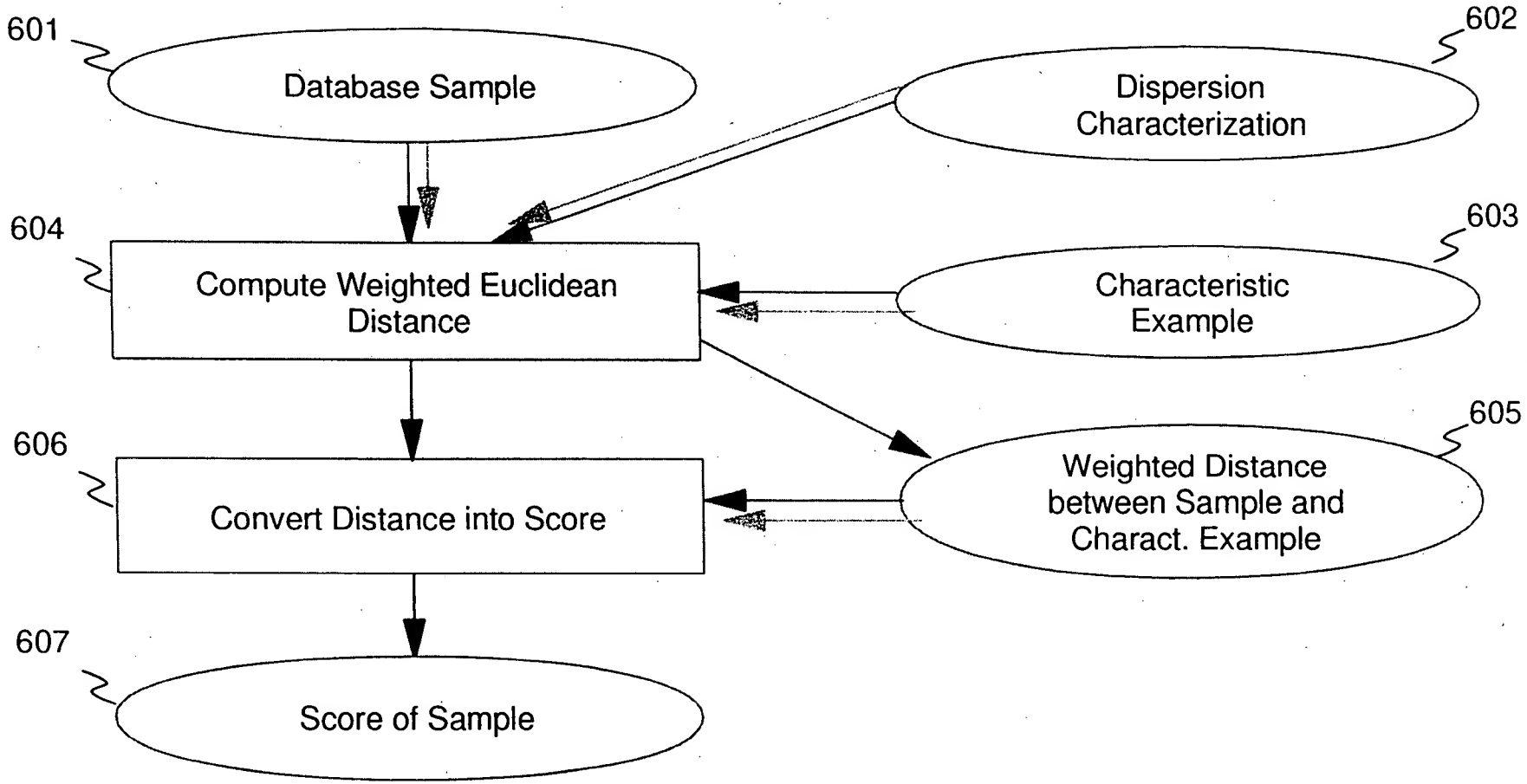

601
Database Sample
602
Dispersion Characterization
604
Compute Weighted Euclidean Distance
603
Characteristic Example
606
Convert Distance into Score
605
Weighted Distance between Sample and Charact. Example
607
Score of Sample

Figure 7

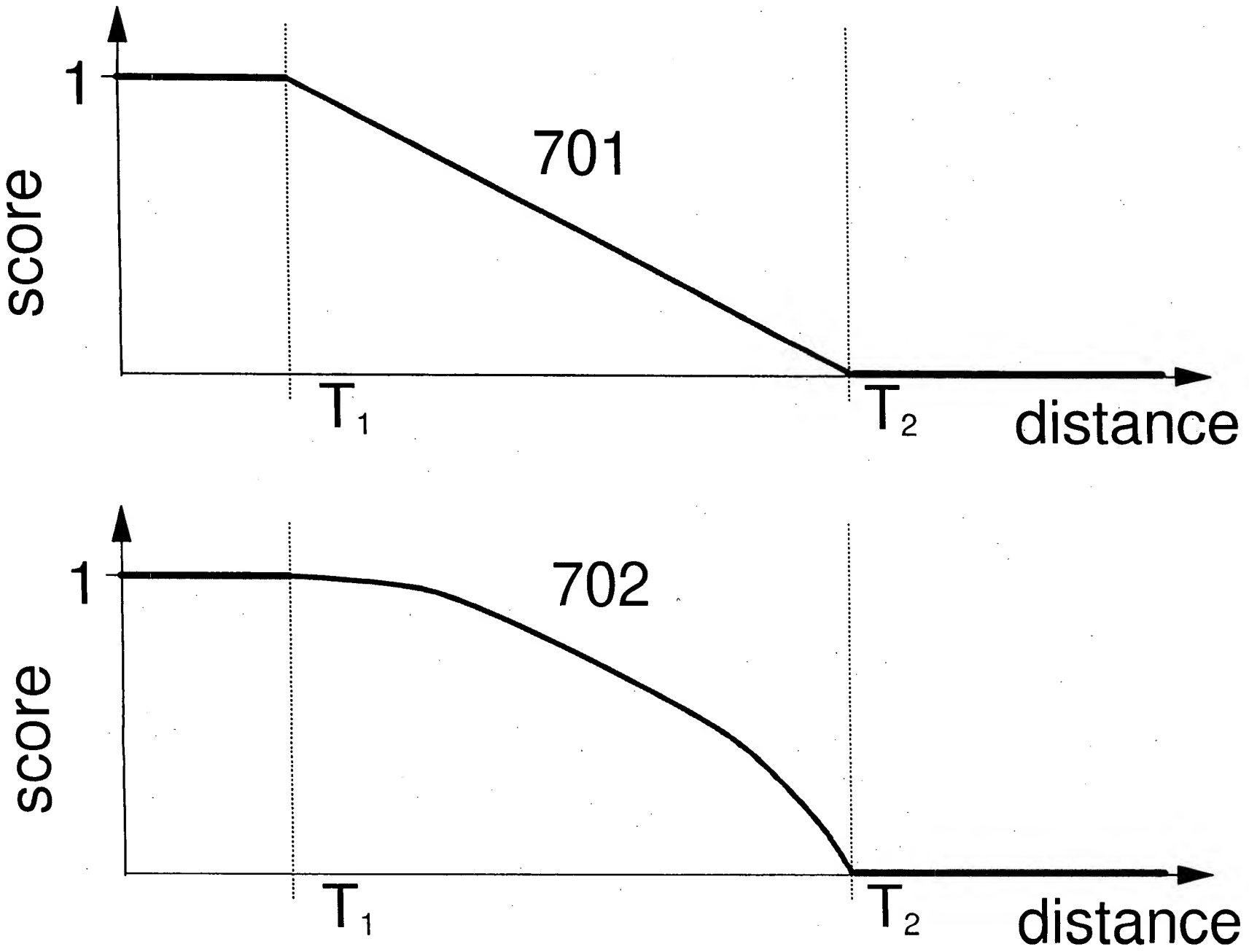


Figure 8

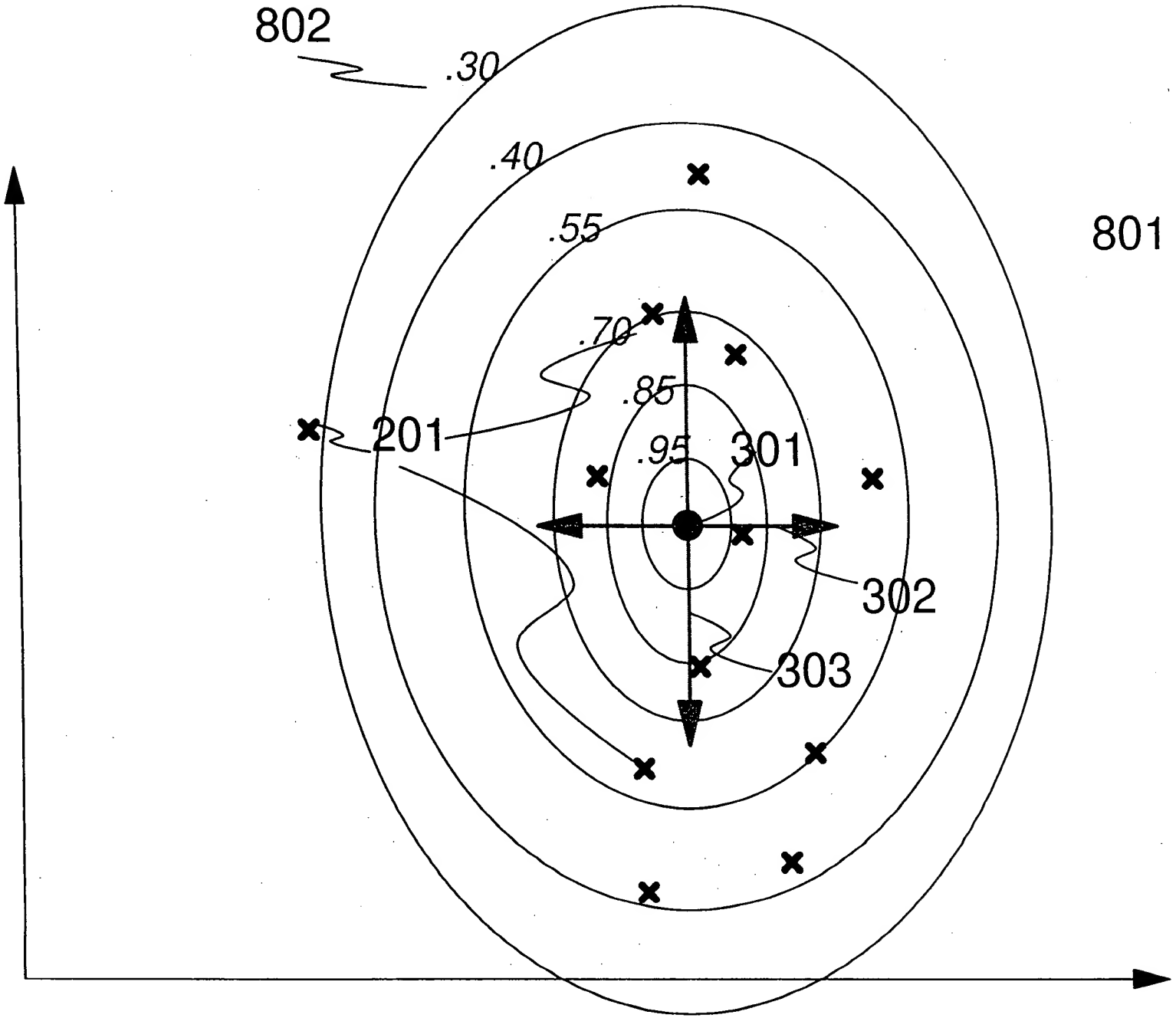


Figure 9

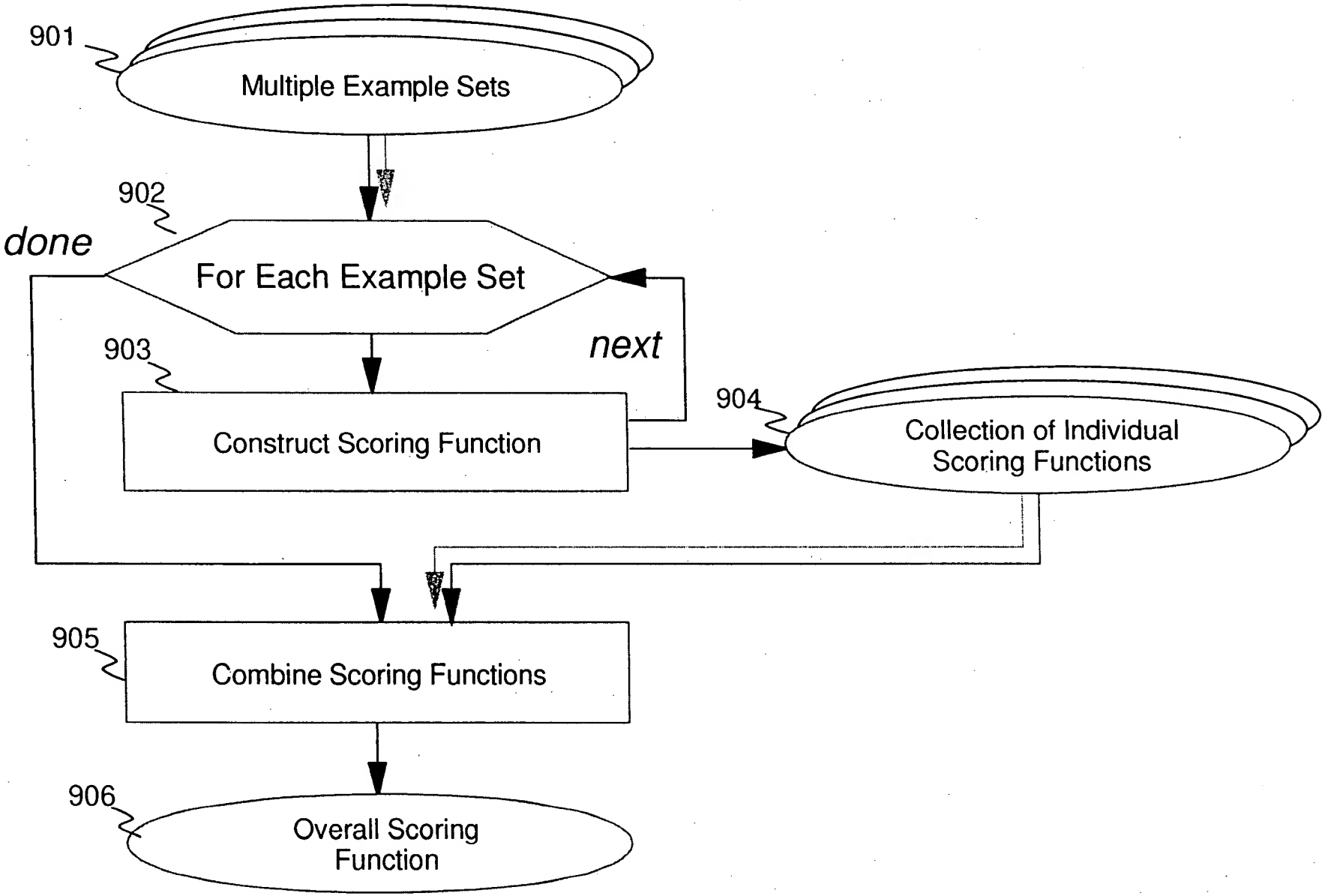
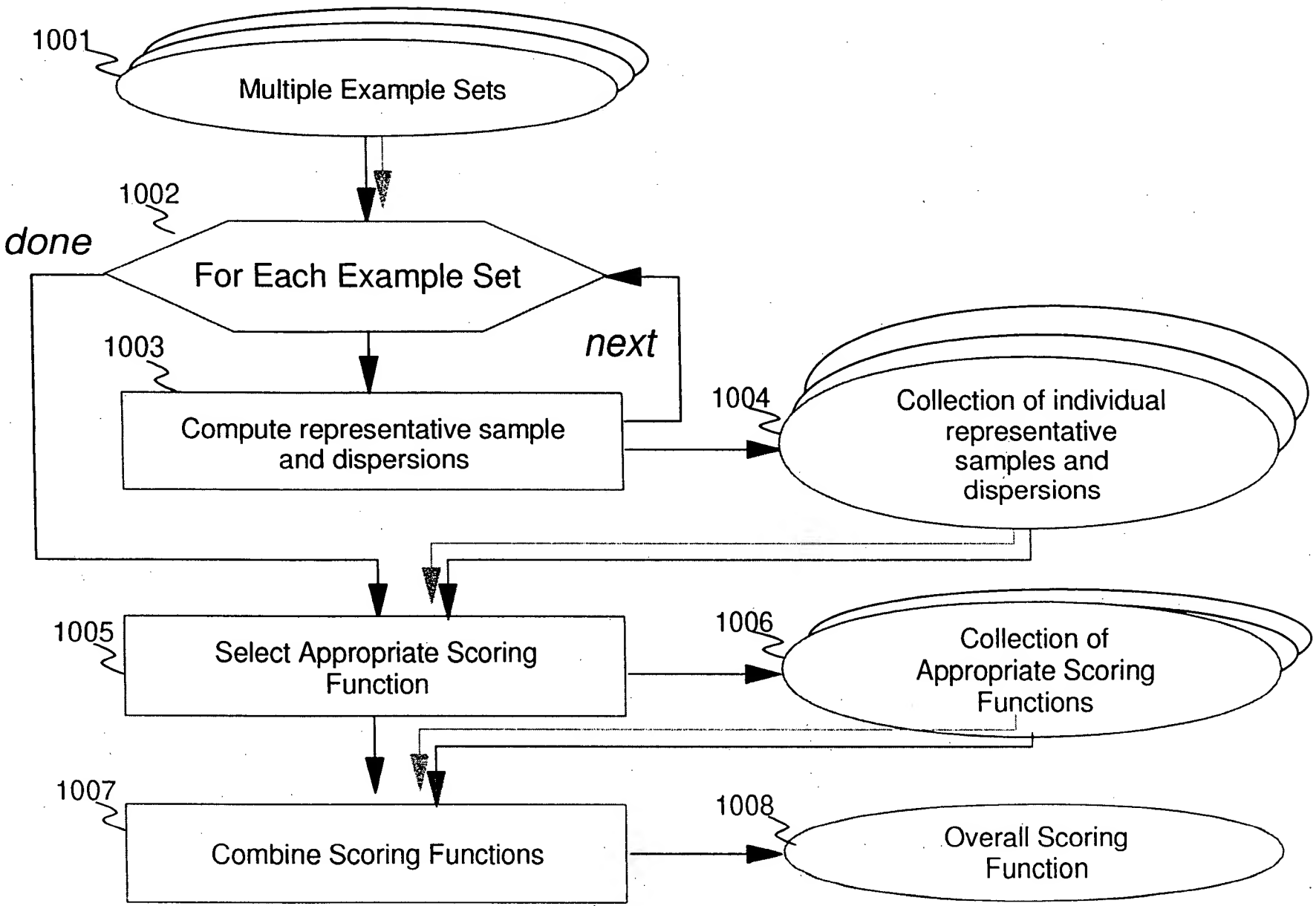
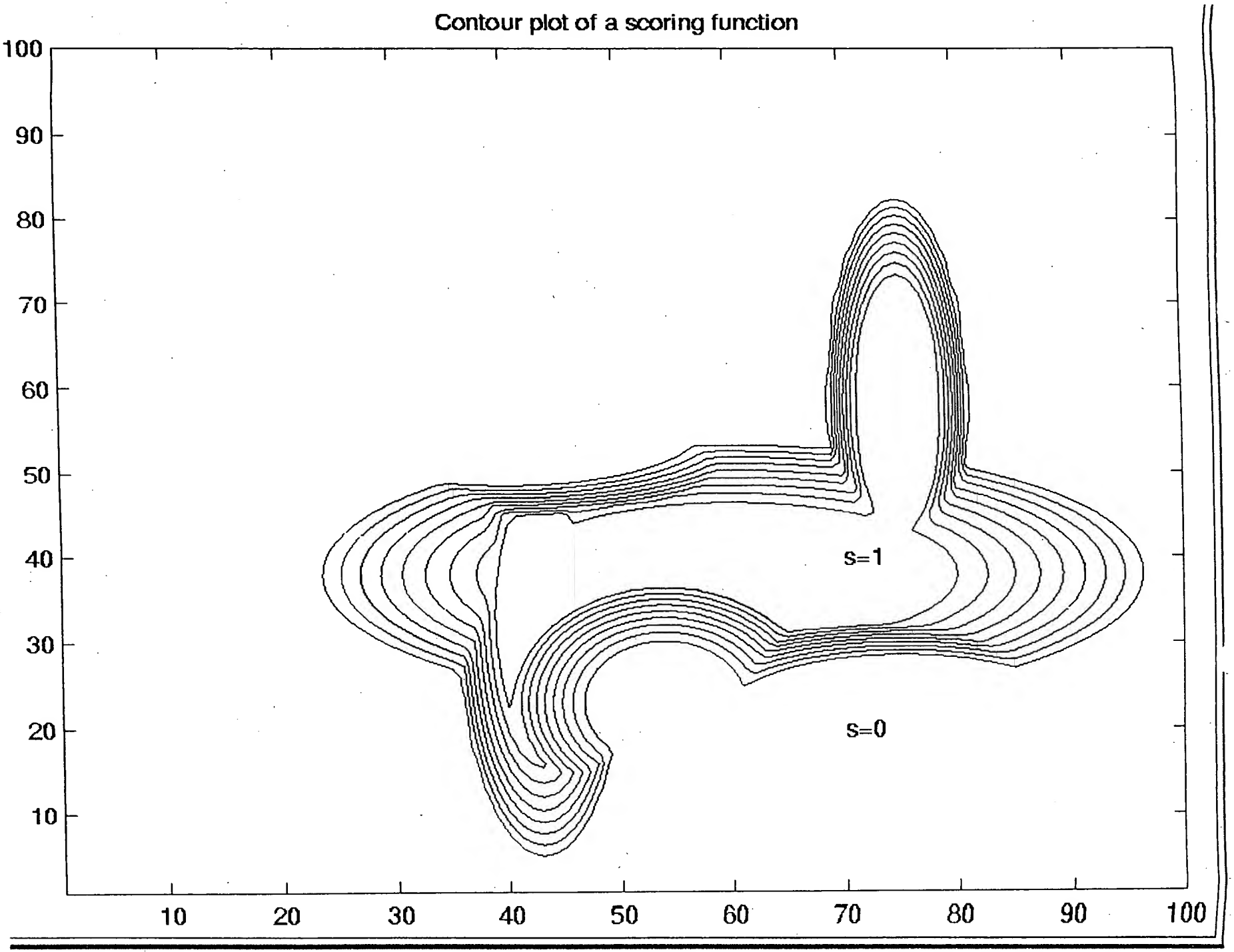
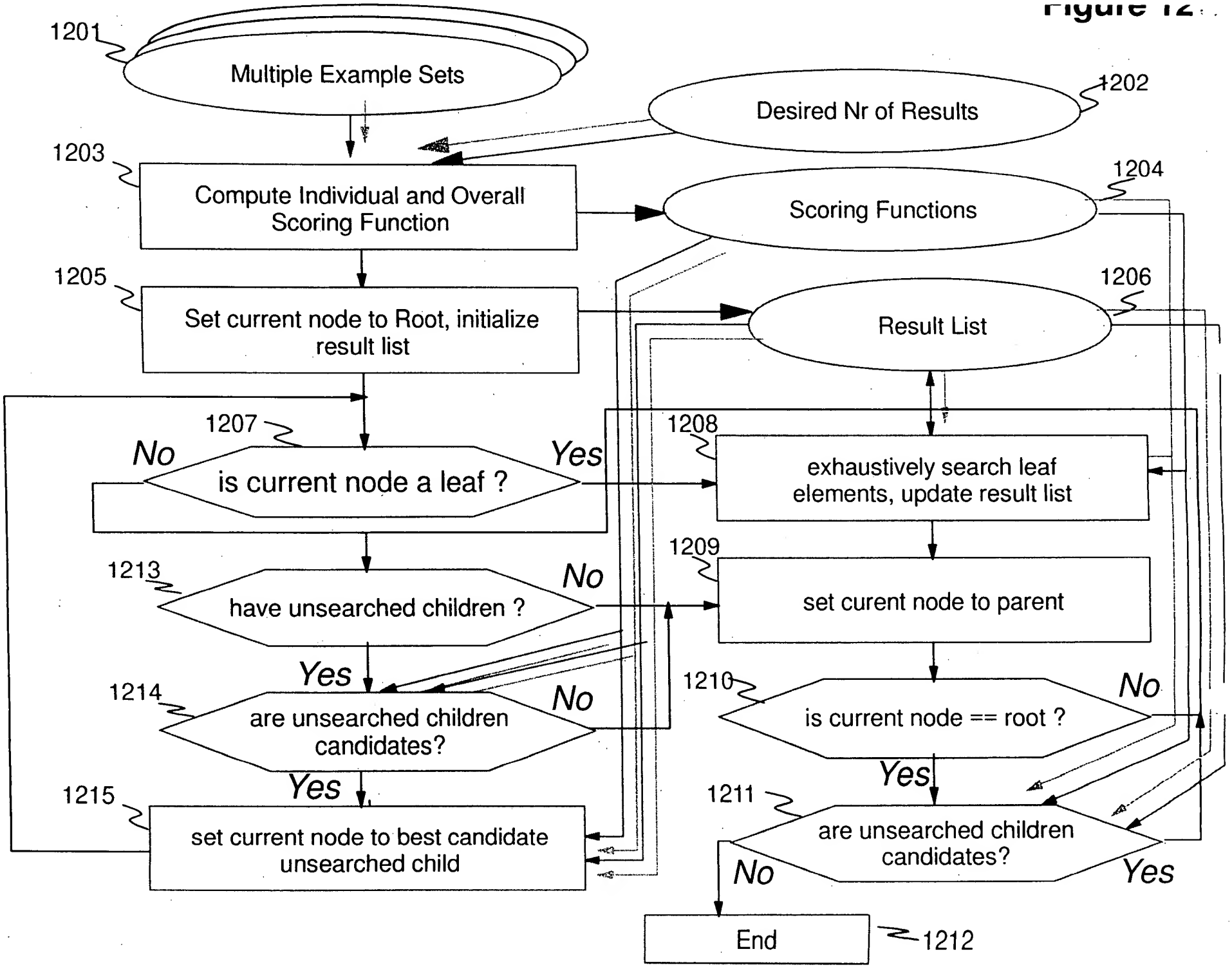

901
Multiple Example Sets
902
done
For Each Example Set
next
903
Construct Scoring Function
904
Collection of Individual Scoring Functions
905
Combine Scoring Functions
906
Overall Scoring Function

Figure 10


1001
Multiple Example Sets
1002
done
For Each Example Set
next
1003
Compute representative sample and dispersions
1004
Collection of individual representative samples and dispersions
1005
Select Appropriate Scoring Function
1006
Collection of Appropriate Scoring Functions
1007
Combine Scoring Functions
1008
Overall Scoring Function


Contour plot of a scoring function
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70
80
90
100
s=1
s=0


1201
Multiple Example Sets
1202
Desired Nr of Results
1203
Compute Individual and Overall Scoring Function
1204
Scoring Functions
1205
Set current node to Root, initialize result list
1206
Result List
1207
No
is current node a leaf ?
Yes
1208
exhaustively search leaf elements, update result list
1209
set curent node to parent
1213
have unsearched children ?
No
Yes
1214
are unsearched children candidates?
No
1210
is current node == root ?
No
Yes
1215
set current node to best candidate unsearched child
1211
are unsearched children candidates?
No
Yes
End
1212

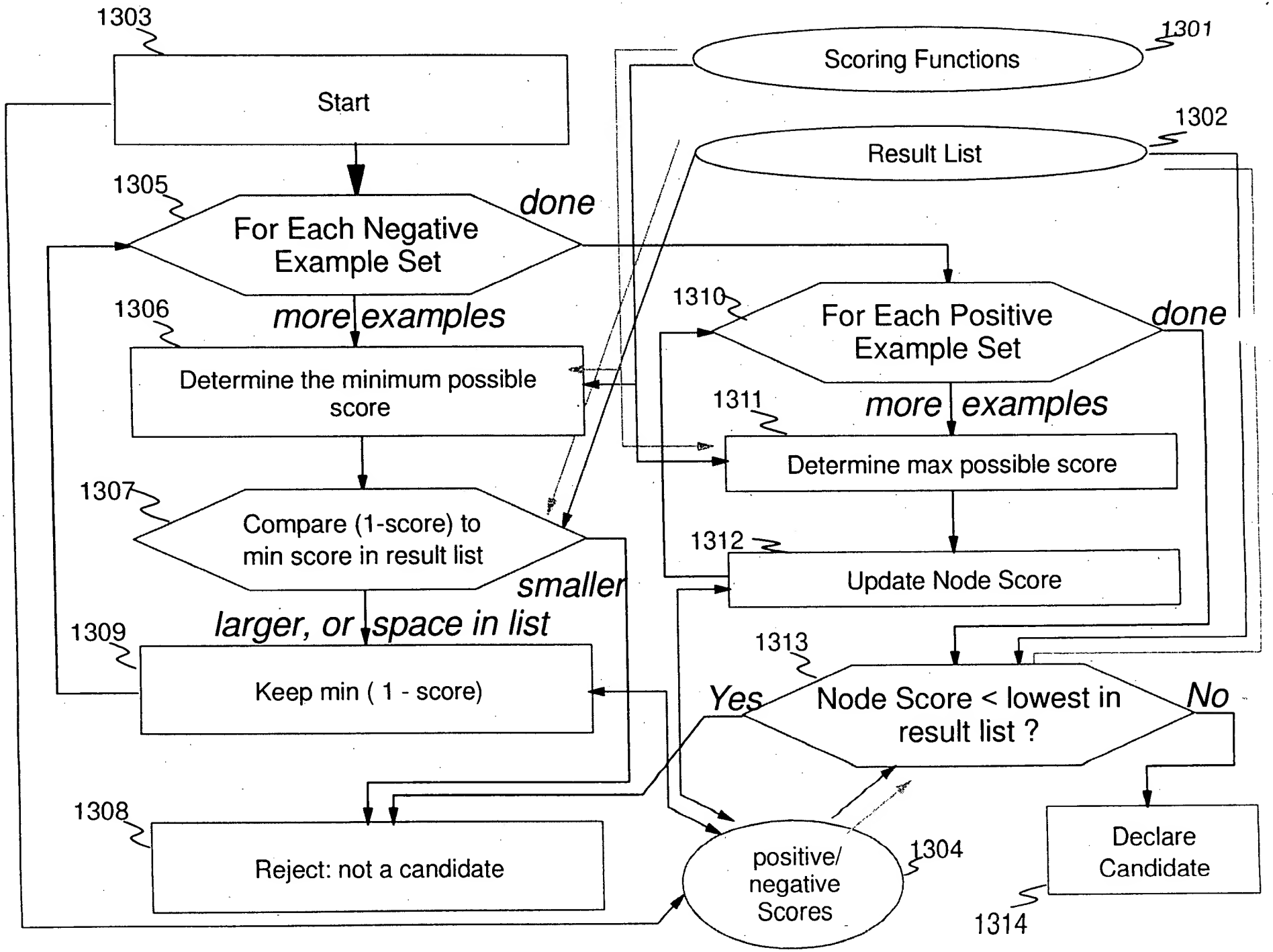

1303
Start
1301
Scoring Functions
1302
Result List
1305
For Each Negative Example Set
done
more examples
1306
Determine the minimum possible score
1307
Compare (1-score) to min score in result list
smaller
larger, or space in list
1309
Keep min (1 - score)
1308
Reject: not a candidate
1310
For Each Positive Example Set
done
more examples
1311
Determine max possible score
1312
Update Node Score
1313
Node Score < lowest in result list ?
Yes
No
positive/ negative Scores
1304
Declare Candidate
1314

Figure 14

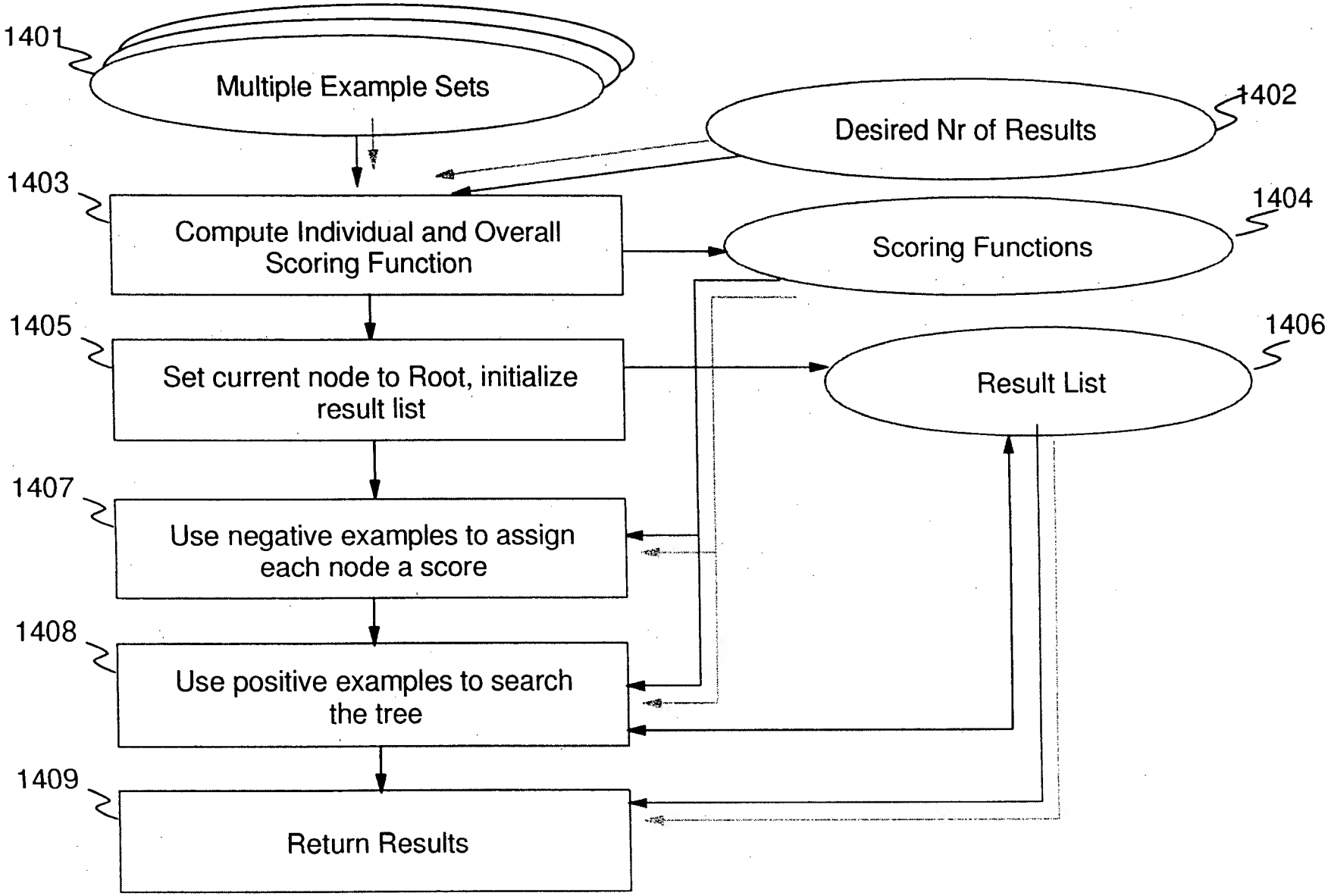
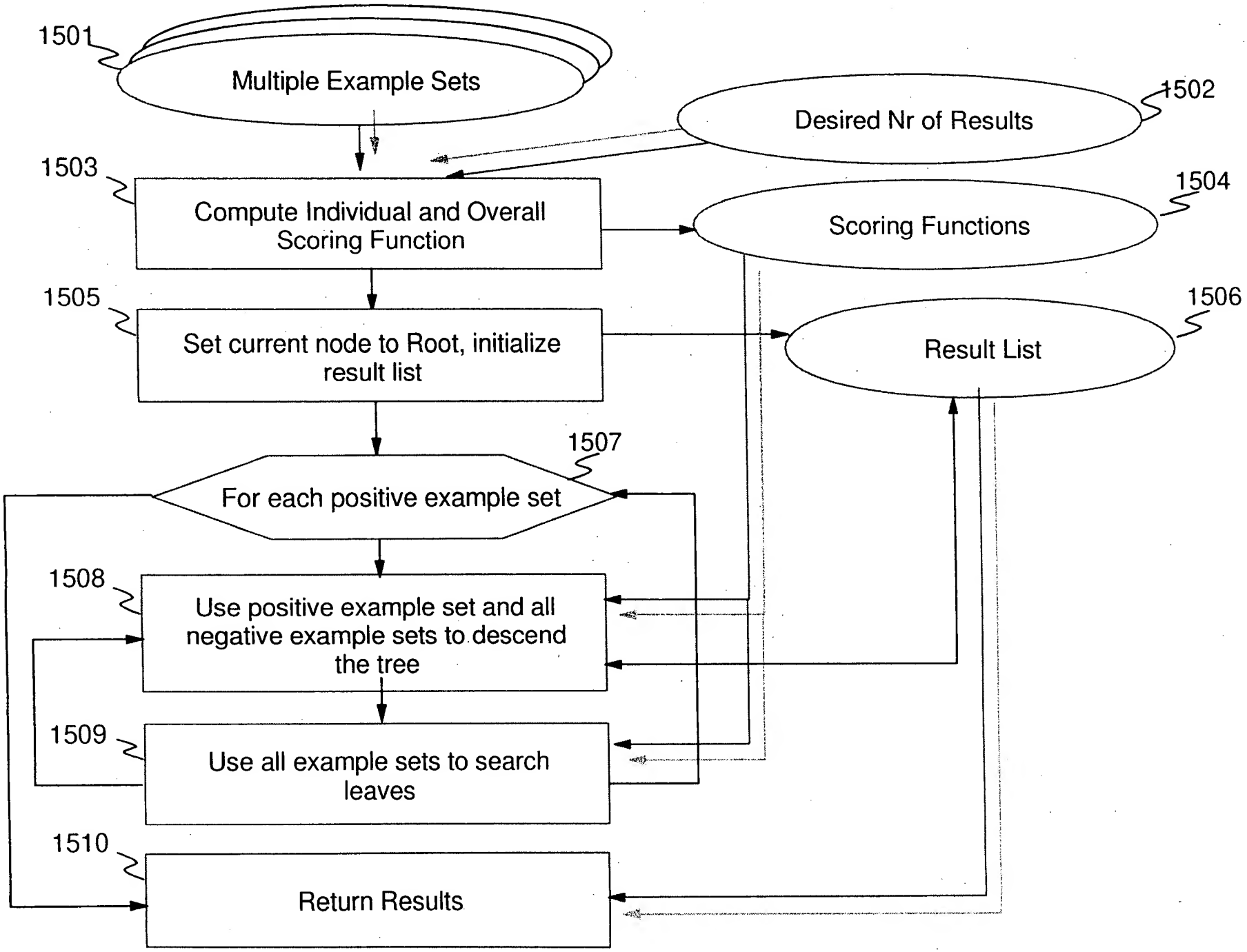

1401
Multiple Example Sets
1402
Desired Nr of Results
1403
Compute Individual and Overall Scoring Function
1404
Scoring Functions
1405
Set current node to Root, initialize result list
1406
Result List
1407
Use negative examples to assign each node a score
1408
Use positive examples to search the tree
1409
Return Results

Figure 15


1501
Multiple Example Sets
1502
Desired Nr of Results
1503
Compute Individual and Overall Scoring Function
1504
Scoring Functions
1505
Set current node to Root, initialize result list
1506
Result List
1507
For each positive example set
1508
Use positive example set and all negative example sets to descend the tree
1509
Use all example sets to search leaves
1510
Return Results

RYAN, MASON & LEWIS, LLP
ATTORNEYS AT LAW
90 FOREST AVENUE
LOCUST VALLEY, NEW YORK 11560
Telephone: (516) 759-2946
Facsimile: (516) 759-9512
Email: wel@rml-law.com

DATE: July 23, 2001	**FILE:** YOR920000742US1

Facsimile Message From: **WILLIAM E. LEWIS**

Please deliver the following pages to:

NAME: Vittorio Castelli
OF: IBM Corporation
FAX NUMBER: (914) 945-4077
NUMBER OF PAGES INCLUDING THIS COVER PAGE: 48
COMMENTS/INSTRUCTIONS:
Dear Vittorio: Please find attached a draft of the patent application relating to your invention. I am still completing the draft claim set and will fax it to you separately by tomorrow afternoon. Please review the attached draft and provide me with your comments at your earliest convenience. Please confirm receipt. Thank you for your assistance and your patience. Best regards,  Bill Lewis

If you do not receive all of the pages, please call us back as soon as possible at (516) 759-2946.